PESQUISA E DESENVOLVIMENTO EM SDN MULTIDOMÍNIO

Desenvolvimento e Implantação dos Controladores SDN Instalados e Testados no Testbed

M3 - A3.3

Softwarização em Redes Abertas e Desagregadas como Habilitador de Aplicações Inovadoras

Programa OpenRAN@Brasil - Fase 1

Sumário

Lista de ilustrações

Lista de tabelas

Glossário

Acrônimos

4G	Quarta Geração
5G	Quinta Geração
6G	Sexta Geração
API	*Application Programming Interface*
BBSim	*BroadBand Simulator*
BGP	*Border Gateway Protocol*
CLI	*Command-Line Interface*
CU	*Centralized Unit*
CU-CP	*Centralized Unit - Control Plane*
CU-CP	*Centralized Unit - User Plane*
DHCP	*Dynamic Host Configuration Protocol*
DTN	*Data Transfer Node*
DU	*Distributed Unit*
DWDM	*Dense Wavelength Division Multiplexing*
EAPOL	*Extensible Authentication Protocol over LAN*

FTTx *Fiber-to-the-x*

gNB 5G *Base Station*

gRPC *Google Remote Procedure Call*

GUI *Graphics User Interface*

INT *In-band Network Telemetry*

K8s *Kubernetes*

MAC *Mandatory Access Control*

MHO *Mobile Handover*

MLB *Mobility Load Balancing*

NETCONF *Network Configuration Protocol*

NNI *Network-Network Interface*

OAI *Open Air Interface*

ODTN *Open and Disaggregated Transport Network*

OLT *Optical Line Terminator*

ONF *Open Network Foundation*

ONIE *Open Network Install Environment*

ONOS *Open Network Operating System*

ONU *Optical Network Unit*

OSC *O-RAN Software Community*

OVS OpenVSwitch

PCI Physical Cell Identifier

PFCP *Packet Forwarding Control Protocol*

PON *Passive Optical Networks*

PTP *Precision Time Protocol*

QoS *Quality of Service*

RAN *Radio Access Network*

RAN *Radio Access Network*

REST *Representational State Transfer*

RESTCONF *Representational State Transfer Configuration Protocol*

RF Rádio-Frequência

RG *Residential Gateway*

RIC *RAN Intelligent Controller*

RIC *Radio Access Network Intelligent Controller*

RSM *RAN Slicing Management*

RU *Radio Unit*

SDN *Software-Defined Networking*

SLAs *Service Level Agreements*

SMO *Service Management and Orchestration*

SSH *Secure Socket Shell*

UE *User Equipment*

UNI *User Network Interface*

UPF *User Plane Function*

URLLC *Ultra-Reliable and Low Latency Communications*

VLAN *Virtual Local Area Network*

VM *Virtual Machine*

VOLTHA *Virtual OLT Hardware Abstraction*

YANG *Yet Another Next Generation*

1 Resumo

O presente relatório detalha a implantação de controladores de redes definidas por software para os domínios P4, RIC, FTTX, DWDM, além do controlador ONOS operando de forma integrada em multidomínios objetivando, fundamentalmente, a interoperabilidade entre domínios tecnológicos e dispositivos de diversos fabricantes para a desagregação da rede e softwarização de seus componentes, sendo parte constituinte da Meta 3 Atividade 3.3 ”Pesquisar, desenvolver e implantar os controladores e componentes de software em cada um dos domínios tecnológicos“, dentro do escopo do projeto openRAN@Brasil em sua primeira fase de execução.

2 Introdução

No contexto do projeto OpenRAN@Brasil, em que se almeja a softwarização dos componentes, a desagregação de sistemas proprietários e a expansão de sistemas abertos que interoperem entre dispositivos e tecnologias de diversos domínios, explora-se a integração entre os domínios de redes sem fio, óptico e de pacotes, por meio dos controladores de cada domínio. Referente ao domínio sem fio há o RIC (*Radio Access Network Intelligent Controller*) com o intuito de gerenciar redes sem fio simplificando a arquitetura ao adotar o uso de Inteligência Artificial (IA) para aprimorar algumas funcionalidades de monitoramento e controle de recursos de rádio. Quanto ao domínio óptico há duas frentes, sendo uma delas o FTTX (*Fiber to the X*) que visa evoluir a infraestrutura de acesso à rede de dados por meio da fibra óptica softwarizando componentes e agregando hardwares de interfaces abertas. Além disso, tem-se o DWDM (*Dense Wavelength Division Multiplexing*) que tem como objetivo otimizar o sistema óptico de transporte de dados, ao extrair a máxima eficiência da infraestrutura de transporte ao utilizar técnicas de orquestração dessa camada, quanto a multiplexação, comutação, gerenciamento, e supervisão dos canais. Considerando o domínio de pacotes toma-se o ecossistema P4 (*Programming Protocol-independent Packet Processors*) que tem como alvo controlar as UPF (*User Plane Function*) para que a entrega de dados ao usuário não seja afetada conforme a demanda variável por tráfego na rede móvel.

Dessa forma, visando apresentar as atividades realizadas para o desenvolvimento do projeto openRAN@Brasil em fase de "pré-testbed", este relatório foi elaborado, contendo informações a respeito de simulações, com descrições da arquitetura de controle e a topologia implementada, características dos *hardwares* usados para executar

as simulações bem como das máquinas virtuais - *Virtual Machines* (VM) criadas para suportar os servidores com os controladores de cada domínio tecnológico, ou até mesmo com domínios interoperantes. Além disso, são mostrados os softwares usados para viabilizar o controle de tais arquiteturas, bem como possíveis customizações necessárias para que esse software se adeque ao cenário abordado. E por fim, os testes feitos para validar tal estrutura e os resultados obtidos por eles.

Este relatório corresponde ao terceiro entregável do Projeto OpenRAN@Brasil desenvolvido em parceria entre CPQD e Rede Nacional de Ensino e Pesquisa (RNP), com auxílio da Universidade Federal do Pará (UFPA) e Universidade de Campinas (UNICAMP), referente a estrutura deste documento, é explicitado no Capítulo 3 o Objetivo deste relatório. No Capítulo 6 são apresentados os resultados obtidos nas simulações feitas conforme investigações relacionadas ao Domínio P4, detalhando a plataforma usada para que haja a interação entre as diversas camadas desse Domínio. No Capítulo 5, são examinadas duas plataformas disponíveis para operação no Domínio RIC, devido suas relevâncias quanto ao ecossistema ao qual o projeto está inserido e as facilidades disponíveis por elas. Na sequência, no Capítulo 7 é averiguado o uso do principal projeto de código aberto disponível para o Domínio FTTX, o qual atende ao requisito de interoperabilidade do OpenRAN@Brasil, possibilitando a desagregação entre fornecedores. O Capítulo 8 contém estudos feitos no cenário do Domínio DWDM, elucidando interações entre a plataforma desenvolvida e disponibilizada pela ONF e os equipamentos que comporão o "testbed" definitivo neste projeto. O Capítulo 9 trata-se do controlador ONOS e das nuances de ajustá-lo para interoperar entre os Domínios P4, DWDM e FTTX. Por fim, no capítulo 10 as conclusões a respeito do que foi apresentado neste relatório e alcançado nessa fase de "pré-testbed".

3 Objetivo

Este relatório corresponde ao entregável da Atividade **3.3 Pesquisar, desenvolver e implantar os controladores SDN e componentes de software em cada um os domínios tecnológicos que compõem o testbed**. Nesta atividade foram implantados os controladores SDN (*Software Defined Network*) dos domínios tecnológicos em um ambiente de "pré-testbed" na generalidade por meio de simulações. Foram testadas as implementações dos controladores, protocolos, interfaces e versões que serão usados para a construção do "testbed".

É importante ressaltar que devido ao atraso na entrega dos equipamentos e componentes para montagem do "testbed" definitivo com equipamentos de fato, resultados a respeito da validação das simulações feitas com a arquitetura de "pré-testbed" ainda não puderam ser realizadas, de forma que essas informações estarão contidas em uma nova versão desse relatório.

4 Domínio RAN

Neste capítulo, é apresentada uma descrição breve das atividades feitas nas etapas de pré-testbed no domínio de RAN (*Radio Access Network*), focadas em estudos, instalação do software referente à pilha de protocolos 5G OpenRAN e integração com RU (*Radio Unit*) comercial.

4.1 "Pré-testbed" - Montagem de setup provisório de demonstração

4.1.1 Descrição da arquitetura de controle

A arquitetura é baseada na desagregação dos componentes que tradicionalmente compõem uma RAN tradicional e na abertura de interfaces entre os próprios componentes [da RAN] e também para outros componentes da rede Open RAN. Os componentes da rede de acesso desagregada se tornam, então, independentes e trabalham em conjunto utilizando as interfaces abertas para prover a mesma funcionalidade que seria entregue com uma rede de acesso monolítica. Ademais, cada componente da rede possui interfaces abertas com outros elementos que não são do domínio RAN para funcionalidades de orquestração e controle inteligente da RAN, por exemplo.

No caso trabalhado neste relatório e no projeto, a RAN é desagregada e passa a ser composta por três componentes:

- CU (*Centralized Unit*): A unidade centralizada da RAN executa as funções de camadas mais altas da RAN, como comunicação com o *Core* da

rede, construção de mensagens de sinalização e controle de tráfego. Por possuir os componentes de camadas mais altas da RAN, não possui requisitos notáveis de latência e capacidade de tráfego na interface com a DU e, por isso, pode ser posicionada distante do local de implantação dos outros elementos da RAN, como em um *datacenter*. A CU pode também ser dividida em CU-CP (CU - *Control Plane*, que executa as funções de camada de controle da CU) e CU-UP (*CU - User Plane*, que cuida do plano de usuário da CU);

- DU (*Distributed Unit*): A unidade distribuída da RAN executa funções intermediárias da rede de acesso, como escalonamento de recurso de rádio para os usuários e uma parte do processamento digital de sinais para as amostras de rádio. A DU, por ser um elemento intermediário na rede, possui interfaces com a CU e a RU. A interface com a RU é restrita em termos de latência aceita e demanda um enlace com alta capacidade. Por esse motivo, a DU precisa estar posicionada próxima à unidade de rádio, no também chamado *edge* da rede;
- RU: A unidade de rádio é o ultimo estágio da desagregação de RAN e executa, no sentido do enlace de descida, a etapa final do processamento digital de sinais e envio das amostras de RF (Rádio-Frequência) por uma antena própria ou anexada.

A Figura 1 mostra os elementos e a conectividade do domínio de RAN em um ambiente OpenRAN. As interfaces NG, E1 e F1 comunicam CU com 5G Core, CU-CP com CU-UP e CU com DU, respectivamente, e são especificadas pelo 3GPP. Já as demais interfaces são especificadas pela O-RAN Alliance: a interface E2 comunica os componentes CU e DU da RAN com o Near-Real Time RIC; a interface O1 é utilizada por todos os componentes do domínio RAN para comunicação com o orquestrador; e, por fim, a interface Open Fronthaul é especificada para comunicação entre a DU e RUs Open RAN.

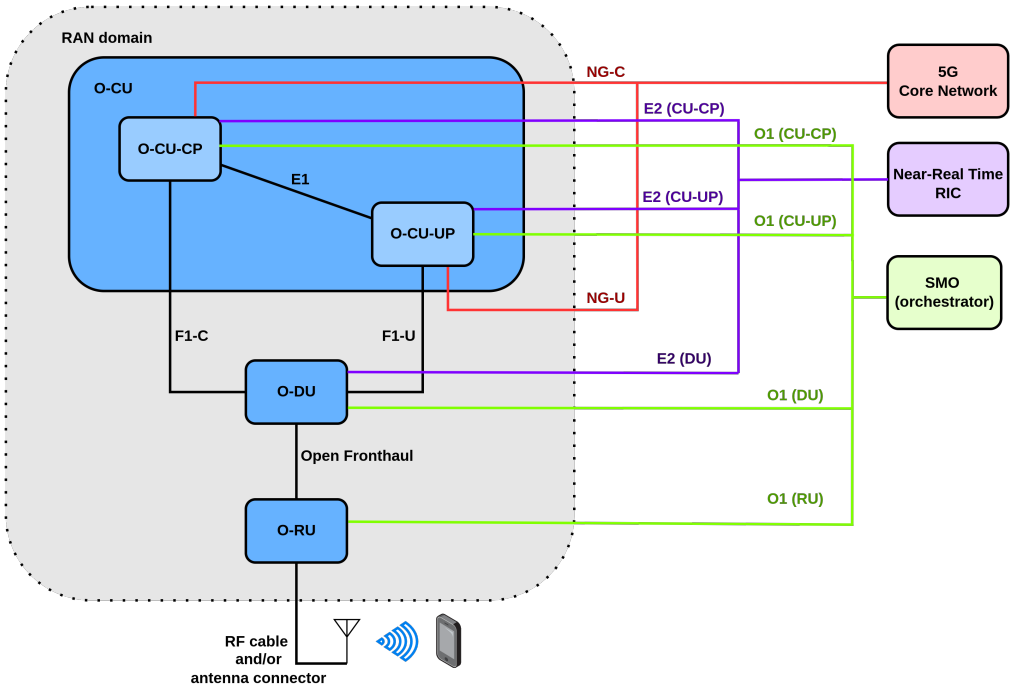


Figura 1 – A conectividade da RAN em um ambiente Open RAN

4.1.2 Descrição da topologia implementada

Devido ao atraso na entrega dos equipamentos e componentes para montagem do "testbed", foi montada uma topologia provisória com equipamentos do CPQD e com alguns equipamentos do projeto que tiveram a entrega feita até o mês de maio. A topologia provisória é descrita na Figura 2 e mostra uma montagem mínima necessária para a implantação de uma célula 5G OpenRAN. O objetivo dessa implantação foi possibilitar a demonstração da rede 5G Open RAN mesmo antes da chegada dos servidores principais do projeto. Alguns elementos da Figura 1 estão ausentes, como SMO (*Service Management and Orchestration*) e RIC, mas estes serão trabalhados e integrados com a evolução do testbed.

Na Figura 2, são mostrados 4 equipamentos principais, sendo dois servidores provisórios para executar a RAN 5G e o Core 5G, um gerador de PTP (*Precision Time Protocol*) e uma RU comercial. Os dois últimos já são equipamento do projeto.

As ligações entre os elementos são caracterizados por 3 planos: plano de

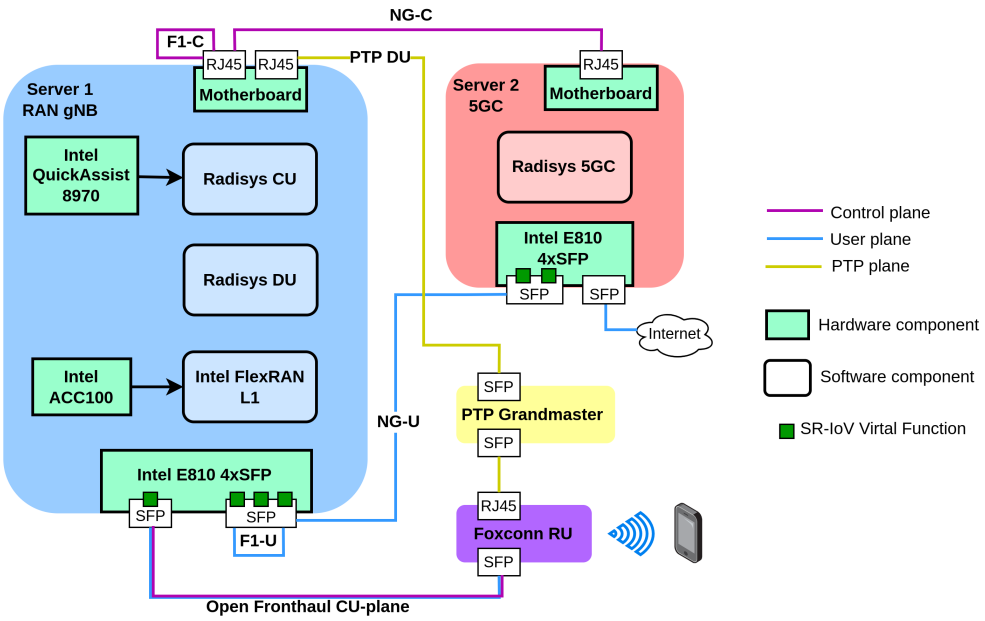


Figura 2 – Topologia inicial montada para testes e demonstração da RAN

Especificações	**Servidor 1**	**Servidor 2**
Modelo Servidor	DELL Poweredge R750	DELL Poweredge R750xs
Processador	2 x Xeon Gold 6348 28 cores	1 x Xeon Gold 5320 26 cores
RAM	8 x 32GB - 256GB	4 x 32GB - 128GB
Armazenamento	2 x 480GB SSD	2 x 480GB SSD
Rede	Intel E810-XXVDA4	Intel E810-XXVDA4
Placas aceleradoras	Intel QuickAssist 8970 Intel vRAN Accelerator ACC100	

Tabela 1 – Especificação de hardware dos servidores provisórios do domínio RAN

controle, utilizando cabos Ethernet de 1Gbps; plano de usuário, com cabos DAC de 10Gbps; e plano PTP, que também utiliza cabos Ethernet de 1Gbps. A conexão do Servidor 1 com a RU é uma exceção, pois plano de controle e usuário estão conectados por um único cordão óptico de 10Gbps. Algumas interfaces dos servidores possuem SR-IoV habilitado.

4.1.3 Características do hardware utilizado

A Tabela 1 mostra as especificações técnicas dos servidores provisórios utilizados na montagem do setup.

Adicionalmente, foi utilizado o switch e PTP grandmaster Falcon-RX/812/G como gerador de PTP para o plano de sincronismo da interface Open Fronthaul e uma RU Foxconn modelo RPQN 7801E, que opera entre 3.7 e 3.8GHz (banda 78 do 5G), com 100MHz de largura de banda. Os smartphones utilizados nos testes do pré-testbed são o Samsung Galaxy Tab S8 Plus e o Samsung Galaxy S22.

4.1.4 Sistemas e módulos de software instalados

Todas as funções da RAN 5G, exceto as executadas pela RU comercial Open RAN, são desempenhadas por módulos de software que foram instalados nos servidores. Os módulos de software são listados abaixo:

- Radisys 5G Core: Core 5G fornecido pela Radisys, versão 4.0.0, instalado e implantado em diversos contêineres *docker*;
- Radisys 5G CU: CU 5G Radisys, versão 4.0.2, instalada em bare-metal;
- Radisys 5G DU FlexRAN: DU 5G Radisys integrável com a L1 Intel FlexRAN, versão 4.0.2, instalada em bare-metal;
- Intel FlexRAN L1: L1 da Intel responsável pelo processamento de mais alto nível de camada física (*High PHY*), versão 22.11, instalada em bare-metal;

Todos os módulos de software são executados em conjunto para a plena implantação da célula 5G. As configurações dos módulos que rodam em bare-metal foram feitas manualmente via edição de arquivos de configuração

4.1.5 Demonstração realizada

Nos dias 22 e 23 de maio de 2023 ocorreu o 24º Workshop da RNP em Brasília e o setup pré-testbed descrito nas sessões 4.1.2, 4.1.3 e 4.1.4 foi levado como demonstração do projeto OpenRAN@Brasil. A demonstração foi um sucesso e mostrou

Figura 3 – Setup pré-testbed montado para o 24º Workshop da RNP (WRNP) em Brasília.

a capacidade do sistema 5G Open RAN em plena operação, alcançando taxas de dados superiores a 1Gbps. A Figura 3 mostra o setup pré-testbed montado no *stand* do evento.

5 Domínio RIC

Neste capítulo, apresentamos uma descrição breve das implementações feitas nas etapas de pré-testbed presentes na Seção 5.1, focadas em estudos e experimentos.

5.1 "Pré-testbed" - Simulações

Nesse ambiente de pré-testbed, estudos foram realizados com duas das principais plataformas RIC (*RAN Intelligent Controller*) open source disponíveis, os RICs das comunidades OSC (*O-RAN Software Community*) e ONF (*Open Networking Foundation*). Como os dois possuem diversos casos de uso disponíveis para testes e são de interesse de todo o ecossistema Open RAN, ambos foram considerados nessa etapa, visto que as instalações da solução de RAN adquiridas pelo projeto ainda não haviam sido realizadas para balizar os testes da equipe.

5.1.1 Descrição da arquitetura de controle

A arquitetura é baseada em microsserviços, cada qual responsável por uma funcionalidade. Pode-se interpretar tal plataforma como um conjunto desses microsserviços, que interagem através de protocolos de comunicação internos. Tal plataforma é implementada sobre uma infraestrutura de conteinerização, com uso de Kubernetes. Com isso, os microsserviços são implementados em pods, que expõem endereços e portas para permitir a coordenação das atividades entre os componentes.

A comunicação entre a plataforma RIC e a RAN se dá através da interface E2, pela qual trafegam os dados de interesse das aplicações do RIC (chamadas de xApps),

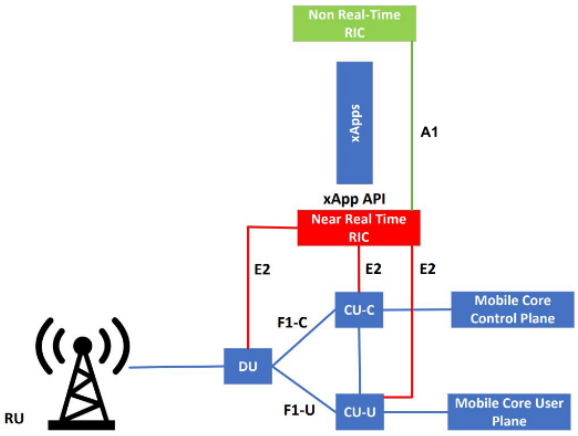


Figura 4 – A conectividade do RIC em um ambiente Open RAN

bem como as mensagens de ação e controle do RIC, visando alguma otimização ou melhoria contínua na RAN.

Os xApps são responsáveis por hospedar algoritmos de otimização, automação, detecção de falhas, entre outras funções, de modo a tornar a RAN mais eficiente. Eles também são tratados como pods no contexto do cluster Kubernetes que contém a plataforma RIC.

A Figura 4 demonstra a inserção da plataforma RIC em um ambiente Open RAN, integrado aos demais elementos através das interfaces já mencionadas. Já na Figura 5, pode-se visualizar a arquitetura interna de um RIC, com seus componentes internos e suas interfaces.

5.1.2 Descrição da Topologia implementada

Como dito na abertura desse capítulo, foram considerados, nessa etapa de avaliação, duas soluções RIC: OSC e ONF. As duas foram implementadas de maneira similar. Foram realizados estudos instalando-as em máquinas virtuais, conectando as plataformas a simuladores compatíveis com cada uma.

Além disso, o RIC ONF permite uma integração com a pilha experimental da OAI, utilizada nessa etapa como teste de ensaio visando a integração com a pilha

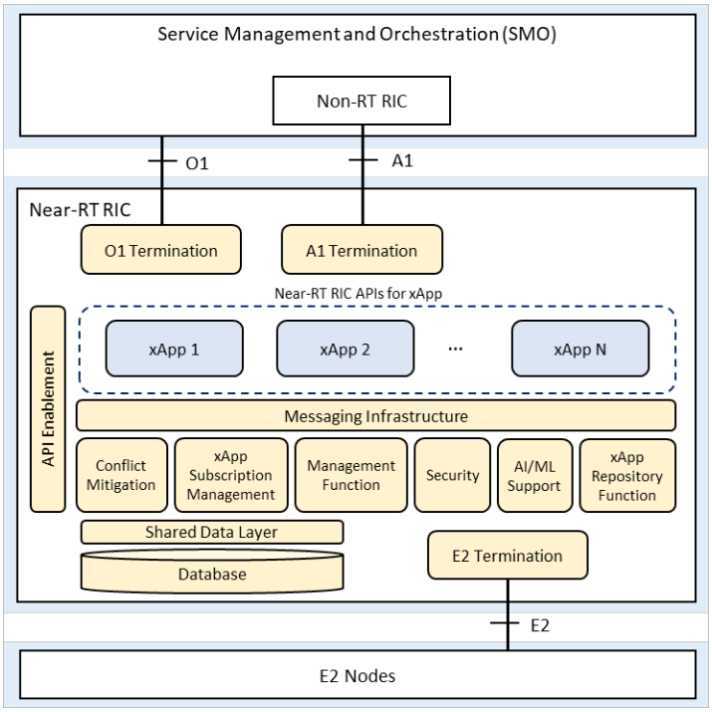


Figura 5 – Os componentes internos e as interfaces do RIC [1]

comercial adquirida pelo programa. Tal teste consiste em conectar o RIC ONF a um notebook, que por sua vez se conecta a um rádio definido por software e a um celular comercial. Tal experimento permitiu verificar, por exemplo, testes de vazão e adaptação de banda (fatiamento de recursos da RAN).

5.1.3 Características do Hardware utilizado

Os componentes de hardware utilizados nessa avaliação são os envolvidos nos testes do RIC ONF com a pilha OAI: notebook, rádio definido por software e um celular comercial.

O notebook empregado tem as seguintes características: sistema operacional Ubuntu Desktop 20.04, memória RAM de 16 GB, 500 GB de disco rígido e processador Intel i7. Vale destacar que, para a conexão com o rádio definido por software, é imprescindível uma porta USB 3.0. O rádio utilizado é uma USRP Ettus do modelo B210. O celular comercial empregado foi um Motorola Moto Edge Plus.

5.1.4 Características da(s) VM(s) utilizada(s)

As VMs utilizadas para testes de ambas as plataformas RIC tem as mesmas características: 16 GB de memória RAM, 120 GB de memória para armazenamento em disco rígido, além de contar com CPU Intel Xeon com 8 *cores*.

5.1.5 Sistemas e Módulos de Software instalados

O sistema operacional utilizado no notebook e nas VMs foi o Ubuntu 20.04. Além disso, por serem soluções em ambiente Kubernetes, faz-se necessário o uso das ferramentas Docker, Kubernetes e Helm. As primeiras coordenam os processos relacionados a conteinerização e virtualização. A última oferece uma gestão de arquivos e pacotes para a instalação de aplicações integradas a esse ambiente em Kubernetes.

5.1.6 Sistemas e Módulos de Software desenvolvidos/customizados

Na etapa de pré-testbed, não foram desenvolvidos softwares adicionais para a realização dos testes.

5.1.7 Testes realizados no Pré-testbed

Com os ambientes simulados construídos para cada um dos RICs analisados, e até com o ambiente real com uso de pilha experimental OAI para o RIC ONF, foi possível realizar uma ampla gama de testes. Em resumo, contemplando os dois RICs analisados, todos os testes possíveis com os xApps disponíveis open source foram avaliados.

Como exemplo, podemos citar, para o RIC OSC, citamos os xApps de *Anomaly Detection* (incluindo seus xApps componentes de *Quality Prediction*, *Traffic Steering*), de coleta de métricas via KPIMON e uso do banco de dados InfluxDB. Nas Figuras 6 e 7, é possível visualizar informações coletadas com o uso do xApp de monitoramento de métricas (KPIMON) a nível de UE e a nível de célula.

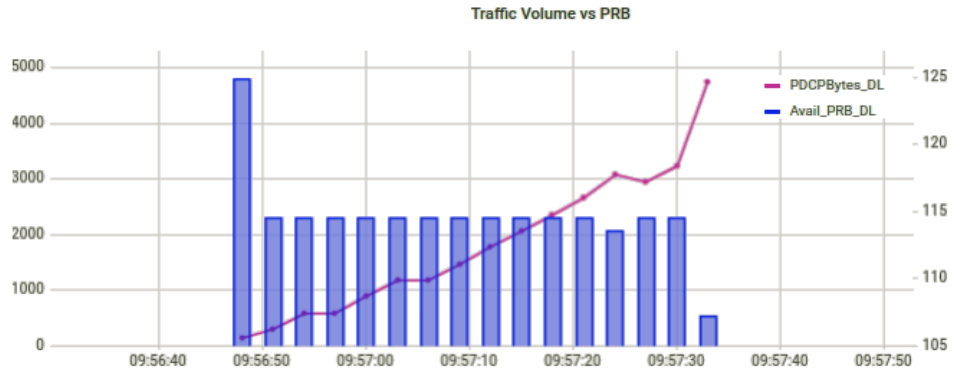


Figura 6 – Tráfego x Nível de PRBs por célula com RIC OSC e xApp KPIMON

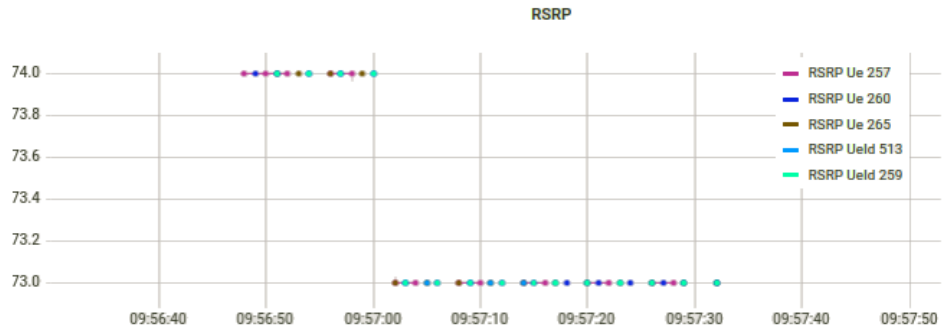


Figura 7 – Medidas de potência para diferentes UEs com RIC OSC e xApp KPIMON

Já para o RIC ONF: as aplicações de *Mobile Handover* (MHO), *Mobility Load Balancing* (MLB), *Physical Cell Identifier* (PCI), sua integração com banco de dados Prometheus e ferramentas de monitoramento Grafana em ambiente Kubernetes, além do *RAN Slicing Management* (RSM). Esse último permite o controle da alocação de recursos da RAN em fatias, de modo a determinar uma extensão da largura de banda disponível em cada uma delas. Consequentemente, as taxas de transferência de dados obtidas em uma conexão também serão afetadas. Nas Figuras 8 e 9, pode-se observar o comportamento descrito, ao adaptar a fatia que atende o UE num teste com *iperf3*.

Figura 8 – Teste de vazão com *iperf3* no ambiente pré-testbed com RIC ONF

Figura 9 – Modificando taxas através de RAN *slicing* com RIC ONF e xApp RSM

6 Domínio P4

Neste capítulo, primeiramente (Seção 6.1) é apresentada uma breve descrição da arquitetura de controle necessária para o SD-Fabric, seguido do detalhamento da topologia e hardwares utilizados no ambiente de testes pré-testbed. Por fim, são apresentados os módulos e softwares instalados para o seu correto funcionamento no ambiente e os testes realizados no ambiente pré-testbed.

6.1 "Pré-testbed" - Simulações

6.1.1 Descrição da arquitetura de controle

SD-Fabric é uma plataforma de código aberto com o objetivo de simplificar e abstrair o plano de dados. Mais especificamente, SD-Fabric é uma malha (*fabric)* de software aberto de *data center* definido totalmente em software. Entre seus principais componentes, podemos citar:

- **Programa P4.** Um programa escrito na linguagem P4 com funcionalidades de encaminhamento básico e recursos avançados, como 5G UPF, INT, *slicing* de rede, etc.
- **Plano de controle SDN.** Um controlador ONOS se comunica com o plano de dados através de uma interface de comunicação com o Stratum no plano de dados.
- **Ferramentas nativas.** Um conjunto de ferramentas nativas preparadas para ambientes de desenvolvimento em nuvem e conceitos de CI/CD (*Con-*

tinuous Integration/Continuous Delivery), que facilitam o monitoramento da pilha completa de produção.

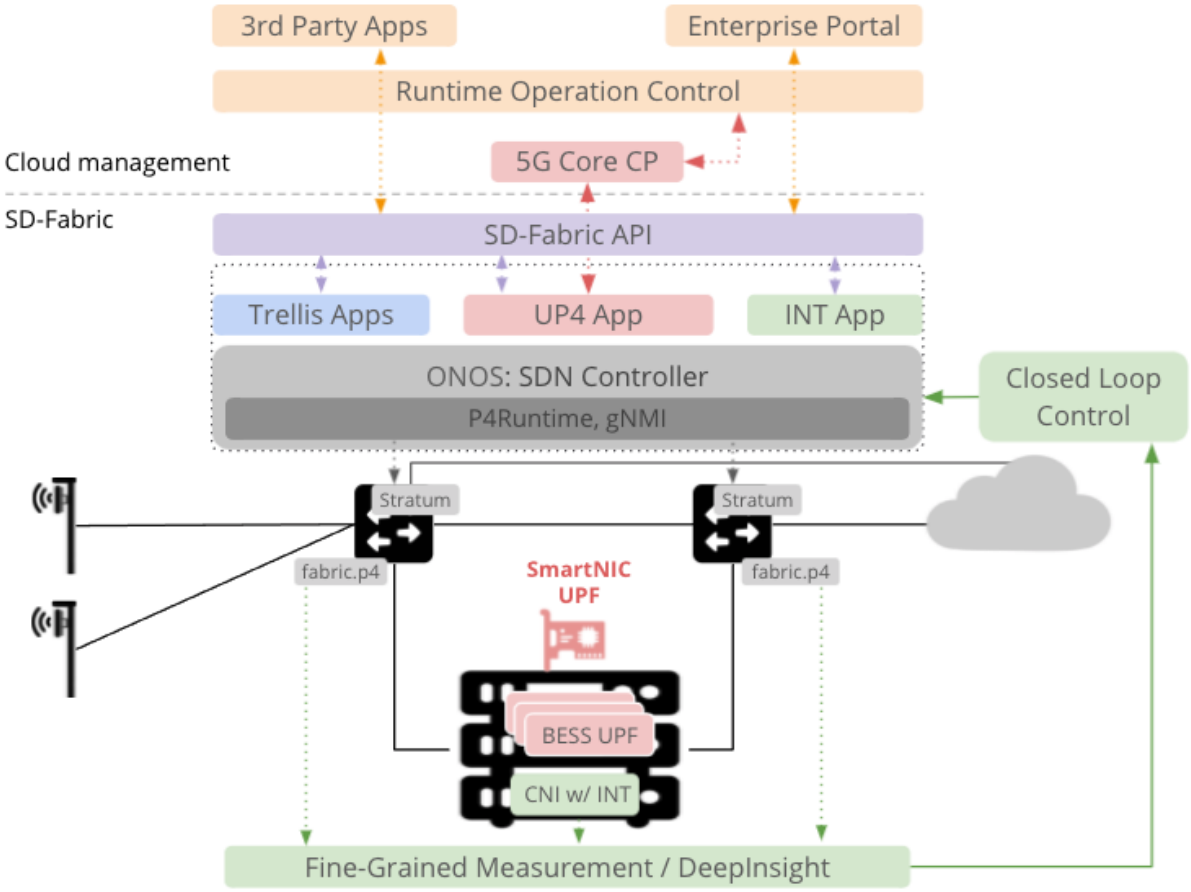


Figura 10 – Visão geral da arquitetura SD-Fabric.

A Figura 10 apresenta a arquitetura proposta pela ONF em maiores detalhes. Visualizando a imagem de uma forma *bottom-up*, os switches programáveis (e.g., Wedge100BF-32X) executam um programa P4 com diferentes funcionalidades (e.g., UPF, *slicing* de rede). Esse programa e demais entradas nas tabelas são gerenciáveis a partir do Stratum, que funciona como um "mini" sistema operacional na arquitetura. O Stratum é responsável por gerenciar essas entradas e é visto como um nodo kubernetes em nossos *deployments* no pré-testbed e testbed. Quando as funcionalidades UPF são executadas por um *profile* do código P4 que as suportem, é dito que se está utilizando o P4-UPF. No entanto, se essas funcionalidades forem executadas em algum componente

externo (e.g., máquina virtual, servidor) executa-se o BESS-UPF [1]. Logo acima, é possível observar o plano de controle (parte acinzentada). Utiliza-se o controlador ONOS para se comunicar com a camada do plano de dados, através de uma canal de comunicação do controlador com o Stratum pelo protocolo P4Runtime [2]. Dentre as principais aplicações a se mencionar, tem-se as aplicações do Trellis [3], UP4, e INT. O primeiro é responsável pelas funcionalidades de encaminhamento básico (e.g., *segment routing*, encaminhamento IPv4/6). O UP4 é o "intermediador" entre o plano de controle e plano de dados, abstraindo o plano de dados como um switch apenas – i.e., *one big switch* –, facilitando a instalação de novas regras de encaminhamento conforme necessário. Por fim, temos as aplicações de monitoramento *In-band Network Telemetry* (INT). Com a linguagem P4, é possível obter métricas do plano de dados em granularidade à nível de pacotes, coletando informações preciosas em tempo hábil para garantir *Service Level Agreements* (SLAs) de aplicações exigentes no âmbito de redes 5G/6G – e.g., aplicações *Ultra-Reliable and Low Latency Communications* (URLLC). Além das aplicações, o SD-Fabric disponibiliza uma API que se comunica com as camadas superiores da *stack* de aplicações da ONF – e.g., SD-Core. Mais especificamente, um componente chamado PFCP-Agent é resposável por traduzir as mensagens com protocolo PFCP do SD-Core para o protocolo P4Runtime – utilizado pelos Stratum no plano de dados.

6.1.2 Descrição da Topologia implementada

Na Figura 11 é apresentada uma visão geral sobre a topologia pré-testbed. Para fins de testes de versionamento, foi utilizada uma VM com as características apresentadas na Seção 6.1.3. Esta VM está situada no ambiente do PoP-RJ e possui acesso SSH direto a um switch programável Wedge100BF-32X. Mais especificamente, para ter acesso ao switch, foram criadas três VLANs principais: (i) a VLAN 970 [2] na rede 192.168.110.0/24 é um enlace de 1GB e contém as mensagens de controle entre o Stratum

[1] O BESS-UPF é um switch virtual, e visto como um container na arquitetura do SD-Core, além do nosso escopo.

[2] Por enquanto a VLAN 970 está sendo utilizada para dados de controle e de tráfego de usuário simultaneamente.

e camadas superiores; (ii) a VLAN 1500 é um enlace de 100GB com responsabilidade de lidar com a demanda de tráfego de usuário entre as gNBs e a rede externa/DNN e; (iii) a VLAN 254 permite acesso público SSH para a VM que, por sua vez, está na mesma subrede do switch – acessível apenas pela rede 192.168.110.0/24.

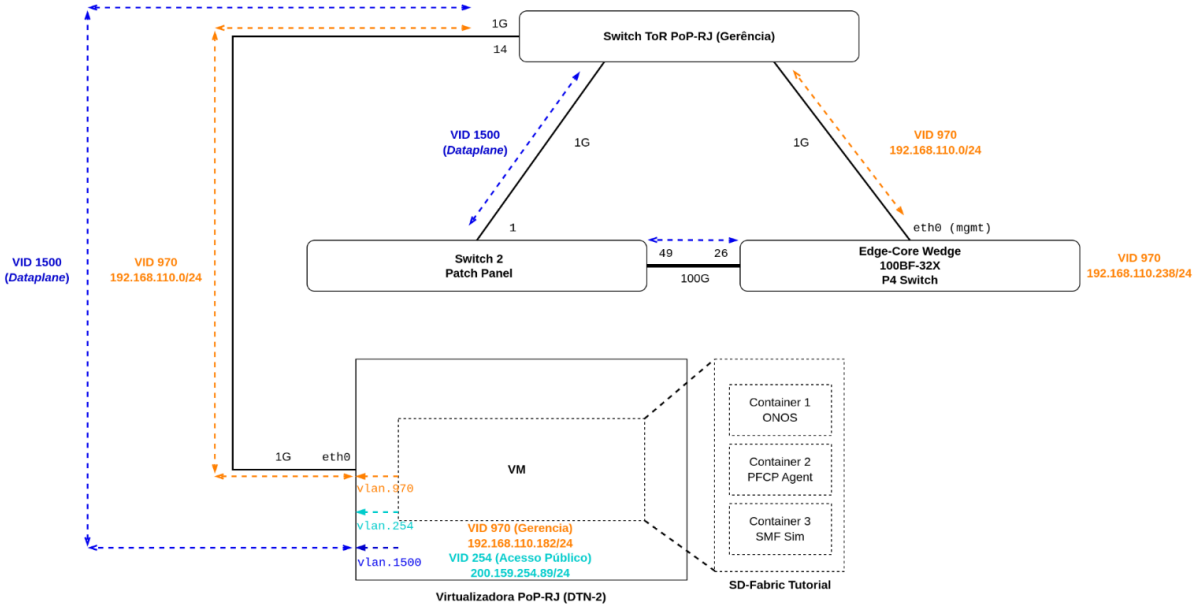


Figura 11 – Topologia de testes.

6.1.3 Características do Hardware utilizado

Na Tabela 2 são apresentadas as principais características físicas do switch Wedge100BF-32X utilizado no ambiente pré-testbed.

Entre os principais recursos disponíveis no switch, podemos destacar:

- Projetado com o silício programável de switch Tofino da Intel Networks.
- Implantação como switch de topo de rack ou spine, suportando 25 GbE para servidores, com uplinks de 40, 50 ou 100 GbE.
- Implantação como switch spine, suportando interconexões de 40, 50 ou 100 GbE para topo de rack e spine.

Specs	
Ports	32
Predominant Port Type	100GbE QSFP28
High Availability	Redundant Power Supply Redundant Fans
Compatible Software Option	Kaloom - Cloud Edge Fabric™ Noviflow - NoviWare Open Compute Project - Open Network Linux SONiC Stratum
Compatible Transceivers	40G QSFP+ 100G QSFP28
Compatible Cables	40G QSFP+ DAC 40G QSFP+ AOC 100G QSFP28 DAC 100G QSFP28 AOC

Tabela 2 – Especificação de hardware do Wedge100BF-32X (i.e., Tofino).

- 32 portas de switch QSFP28, cada uma suportando 1 x 100 GbE, 1 x 50 GbE ou 1 x 40 GbE, ou através de cabos de breakout, 4 x 25 GbE ou 4 x 10 GbE.
- Capacidade de switching de camada 2 ou camada 3 de 6.4 Tbps (duplex completo).
- Suporta corredor quente/frio com fluxo de ar de porta para fonte de alimentação.
- Todas as portas na parte frontal; fontes de alimentação e ventiladores acessíveis na parte traseira.
- Fontes de alimentação CA ou CC de 48V redundantes, hot-swappable e com compartilhamento de carga.
- Módulos de ventilador redundantes 4+1, hot-swappable.

- Switch pré-carregado com diagnósticos e com *Open Network Install Environment* (ONIE) para carregamento automatizado de ofertas de software NOS de código aberto e comerciais compatíveis.
- Compatível com Kaloom Cloud Edge Fabric™.
- Compatível com Noviflow NoviWare.
- Compatível com Stratum.
- Compatível com o software de rede de código aberto SONiC.

6.1.4 Características da(s) VM(s) utilizada(s)

Para implementação do SD-Fabric, é necessário no mínimo uma unidade de processamento, um servidor ou uma máquina virtual com os seguintes requisitos:

- Processador com 8 núcleos;
- 16 GB de memória RAM;
- Disco rígido com capacidade mínima de 50 GB.

6.1.5 Sistemas e Módulos de Software instalados

Por ser uma plataforma de software aberto, o SD-Fabric é, consequentemente, instalado em uma plataforma de software aberto. Neste caso, optou-se por inicialmente realizar os testes no Ubuntu 18.04 e, mais recentemente, no Ubuntu 20.04. Dentre os demais componentes necessários para o correto funcionamento do SD-Fabric, podem ser citados:

- Kubernetes;
- Docker;

- Helm;
- SONiC;
- Calico.

6.1.6 Sistemas e Módulos de Software desenvolvidos/customizados

Nesta seção serão discutidas as principais modificações que foram necessárias para o correto ambiente onde o SD-Fabric foi instalado.

Nomeação do serviço do PFCP-Agent. Um dos problemas encontrados com o PFCP-Agent foi a necessidade de alterar o nome padrão do seu serviço que estava na forma `<namespace>-<pfcp-agent>` para apenas `pfcp-agent`. Dessa forma, o serviço voltava a obter IP e funcionar corretamente - o mesmo ocorre com o seu respectivo pod.

Calico utilizando VXLAN. Por padrão, o Calico utiliza regras BGP para estabelecer conexão entre os nodos. No entanto, com BGP encontrava-se o seguinte erro *"BIRD is not ready: BGP not established..."*, indicando que o componente BIRD não estava presente. Para solucionar isso, optou-se por utilizar VXLAN em todos os domínios. Dessa forma, a conexão voltou a ser estabelecida e os pods afetados foram levantados normalmente.

6.1.7 Testes realizados no Pré-testbed

Devido a problemas de conexão com o Stratum até o momento, que é um componente essencial para a conexão com os switches e seus *profiles* - e.g., P4-UPF, INT - não foi possível estabelecer testes de conectividade GTP-U que seriam realizados com uma VM instanciada com o UERANSIM, um emulador de UE + gNB. Porém, em testes mais recentes, foi possível pela primeira vez identificar o Stratum corretamente utilizando apenas uma instalação com o Docker, isto é, sem os *charts* sugeridos pela ONF no Kubernetes. Veja a Figura 12 a seguir, que ilustra a conexão realizada.

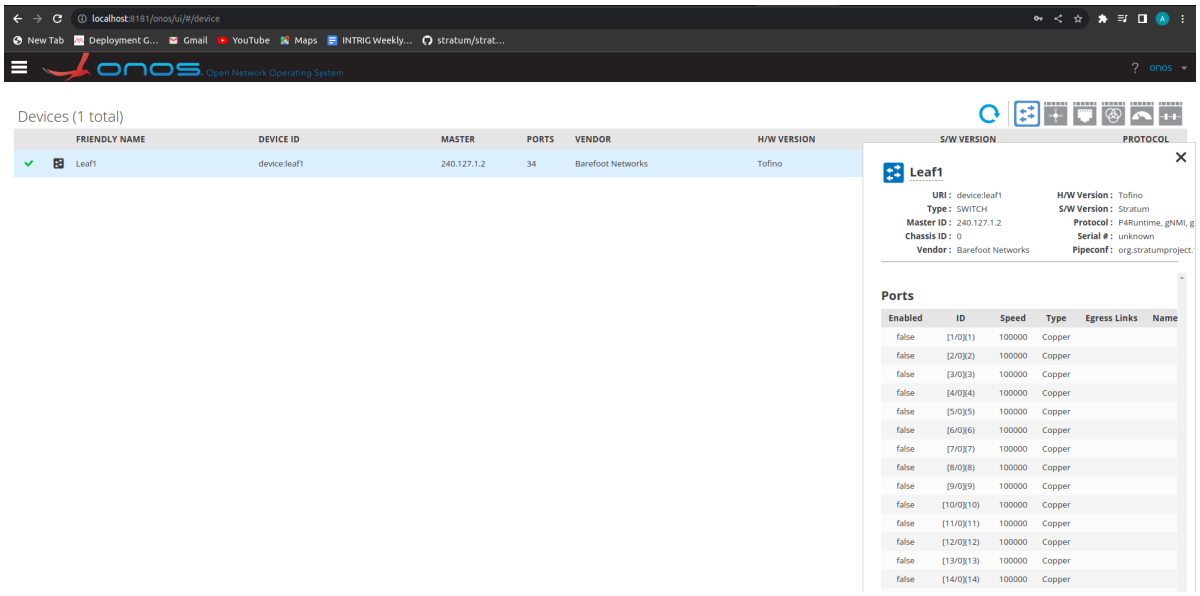


Figura 12 – ONOS identificando corretamente o Stratum - instalação docker.

Como pode ser observado, ao passar o `chassis_config` padrão, todas as portas são reconhecidas corretamente pelo ONOS. Mais especificamente, está sendo utilizado: (i) o `SONiC 20230614` homologado pela ONF para o switch no modelo Wedge 100BF-32X - o mesmo utilizado; (ii) o `Stratum 9.7.0` e; (iii) `ONOS 2.5.7-rc2`.

7 Domínio FTTX

7.1 "Pré-testbed" - Simulações

7.1.1 Descrição da arquitetura de controle

Virtual OLT Hardware Abstraction (VOLTHA) é um projeto de código aberto para criar uma abstração de *hardware* para equipamentos de acesso de banda larga. Atualmente, ele suporta o princípio de desagregação com vários fornecedores. Sumariamente, este projeto é composto por duas camadas principais: 1 - Infraestrutura, onde são armazenados os dados, mensagens e o controlador de SDN; 2 - Pilhas, que incluem o VOLTHA *Core*, adaptadores e o agente *openflow*. Além disso, há uma camada opcional, que gerencia os dispositivos, por exemplo uma atualização de software por APIs (*Application Programming Interfaces*). Na Figura 13, é apresentado o diagrama da arquitetura do VOLTHA, em que cada retângulo representa os contêineres dos componentes, comumente implementados em *clusters kubernetes*. Representando a camada de infraestrutura estão o controlador ONOS (*Open Network Operating System*), o Kafka e o Etcd; enquanto que VOLTHA *core*, OpenOLT, OpenONU Adaptor e Whitebox OLT representam as pilhas do VOLTHA.

7.1.2 Descrição da Topologia implementada

Para viabilizar o *deployment* e testes de validação em um ambiente emulado, foi empregada a topologia apresentada na Figura 14. Nela é apresentado um servidor "VOLTHA" em cada um dos sites, onde são testados todos os recursos relacionados ao SD-PON e VOLTHA. O servidor "VOLTHA" é implementado em uma VM (*Virtual Ma-*

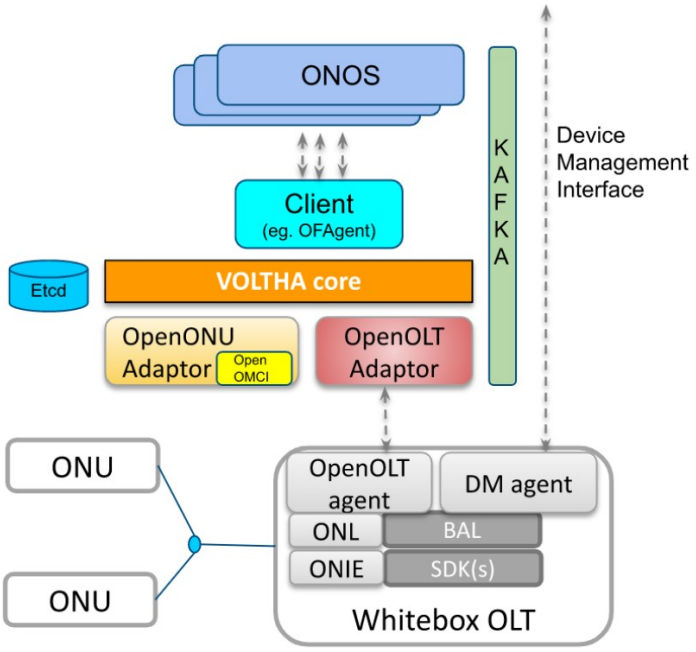


Figura 13 – Caption

chine) com tecnologia de *cloud computing*, mais especificamente *kubernetes*, e a topologia PON, que será implementada com OLTs e ONUs programáveis na versão definitiva do testbed, foi implementada em caráter temporário com o BBSim (*BroadBand Simulator*). Nota-se também na Figura 14 um "*Jump Host*" que apoia a interconexão entre os sites através da criação de VPNs. Por fim, existe o *firewall* usado para a segurança de acesso aos *sites*.

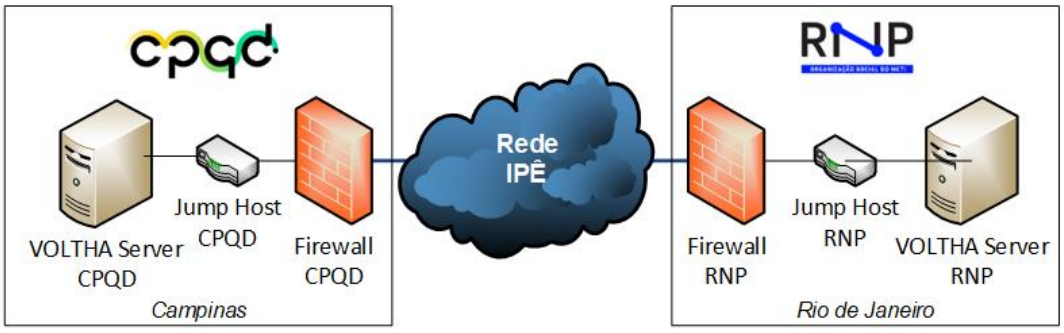


Figura 14 – *Cluster kubernetes* do VOLTHA.

7.1.3 Características do Hardware utilizado

7.1.4 Características da(s) VM(s) utilizada(s)

Para implementação do VOLTHA é necessário no mínimo uma unidade de processamento, um servidor ou uma máquina virtual com os seguintes requisitos:

- Processador com 8 núcleos
- 16 GB de memória RAM
- Disco rígido com capacidade mínima de 60 GB

Idealmente, em ambiente de produção para alta disponibilidade, deve haver 3 servidores.

7.1.5 Sistemas e Módulos de Software instalados

O VOLTHA é baseado em plataformas de código aberto, logo o ambiente de instalação também é baseado em código aberto, como o sistema operacional da Linux Foundation. Portanto, o VOLTHA deve ser instalado em plataforma Linux com o sistema Ubuntu (versão preferencialmente utilizada 20.04). Dentre os componentes de software necessários para a instalação do VOLTHA são:

- Kubernetes
- Docker
- Helm

7.1.6 Sistemas e Módulos de Software desenvolvidos/customizados

A abordagem do sistema de controle e gerência da rede FTTx desagregada e aberta apresenta uma arquitetura baseada em interface de linha de comando e interface gráfica GUI (*Graphics User Interface*), sendo os ambientes utilizados para controle,

monitoramento e operação da topologia virtualizada, gerenciamento de serviços, provisionamento de clientes, entre outros. Dessa forma, o sistema de gerenciamento se torna limitado ao ambiente local de execução e ao uso de interface gráfica para a utilização de certas aplicações, reduzindo a flexibilidade de comunicação com um agente externo ao sistema.

Nesse contexto, desenvolveu-se o módulo de interface de alto nível para orquestrar o sistema de gerenciamento da rede FTTx em ambiente externo à arquitetura do sistema. A arquitetura do módulo é baseada em linguagem de programação de alto nível que expande a capacidade de desenvolvimento do operador, possibilitando o emprego do módulo em servidores externos ao *cluster* de operação do sistema. Na Figura 15, observa-se o módulo de orquestração de alto nível integrado ao sistema de gerenciamento com conexões para os elementos das camadas de controle (ONOS), virtualização (VOLTHA) e encaminhamento (topologia e Kafka), podendo pertencer ao mesmo ambiente de implementação ou em outro ambiente.

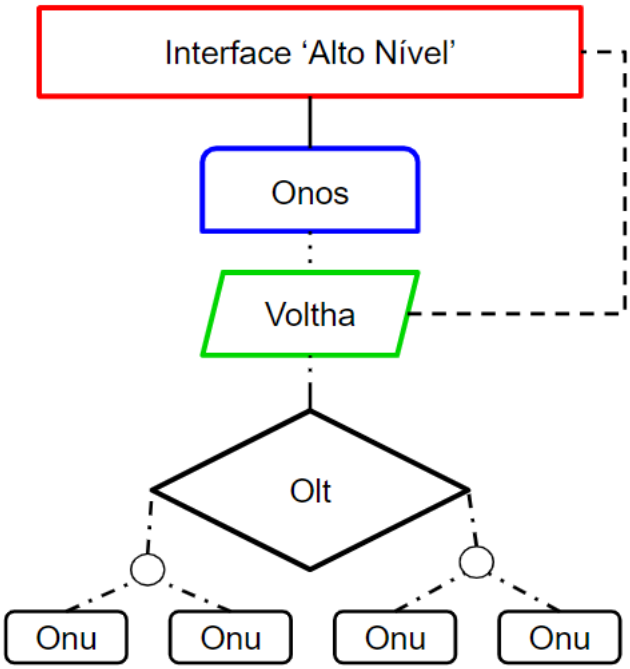


Figura 15 – Módulo de interface de gerenciamento conectado à arquitetura da rede FTTx.

Dentre os componentes do módulo de interface de gerenciamento, encontra-se a divisão em dois protocolos de comunicação: gRPC e REST. O protocolo gRPC abrange os blocos relacionados ao agente de virtualização e que se comunica diretamente com a topologia da rede FTTx. Na comunicação via REST, os componentes são responsáveis pelas aplicações requeridas do controlador e transferir os serviços para o sistema. Dada a Figura 16, tem-se o fluxo de blocos construtivos de um modelo inicial da interface de alto nível para aplicação de um provisionamento de *subscriber* na rede FTTx com os seguinte passos:

- Criação e conexão da OLT virtualizada;
- Verificação do pré-provisionamento da OLT;
- Ativação dos equipamentos da topologia;
- Verificação da topologia virtualizada;
- Criação das configurações de serviços;
- Aplicação das configurações ao controlador;
- Provisionamento do *subscriber*;
- Visualização dos *subscribers* provisionados e operacionais.

De modo complementar, o módulo da interface pode ser incrementado com recursos de monitoramento, alarme, detecção de falhas, entre outros. A flexibilidade de implementação é um grande fator para expandir a capacidade de operação, tornando-se possível a construção de novo blocos e incluí-los na mesma infraestrutura de acordo com os requisitos de desenvolvimento.

7.1.7 Testes realizados no Pré-testbed

O ambiente de implementação do VOLTHA é construído em *cloud* na arquitetura de *kubernetes*. Nesse modelo, as funcionalidades do sistema de virtualização de

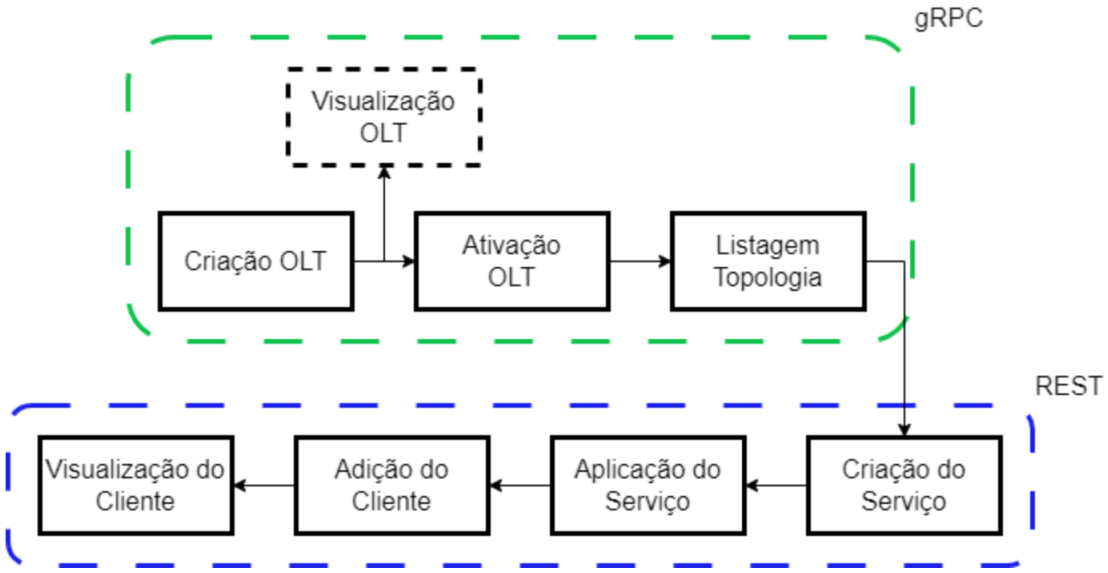


Figura 16 – Camadas de controle e serviço do gerenciador inteligente de rede óptica.

redes ópticas passiva (*Passive Optical Networks* PON) são distribuídas em um *cluster* com camadas de acordo com a estrutura do VOLTHA. O acesso às camadas do cluster são definidas pelas interfaces *northbound* e *southbound*. Tanto a camada de controle quanto a pilha do VOLTHA possuem ambas as interfaces.

Dentre os protocolos para comunicação, podem-se utilizar gRPC para a conexão ao VOLTHA, e openflow para a conexão com o controlador. A CLI (*Command-Line Interface*) é outra forma de acesso ao sistema e também ao controlador. A interface *southbound* do controlador é responsável por conectá-lo à camada do VOLTHA utilizando protocolo openflow. No nível mais elevado, encontra-se a interface *northbound* do ONOS, responsável por receber e executar requisições para o controlador proveniente de um orquestrador ou operador externo, com uso de CLI ou utilizando API REST.

A Figura 17 mostra uma saída do comando "*kubectl*", listando os *Pods* do cluster da infraestrutura implementada do VOLTHA em *kubernetes*. Dadas as interfaces da infraestrutura de controle do SD-PON, neste trabalho será apresentado o uso de REST para o gerenciamento da rede, bem como a CLI do VOLTHA para o controle da

NAMESPACE	NAME	READY	STATUS	RESTARTS	AGE
infra	bbsim-sadis-server-7547584ff6-pjjlm	1/1	Running	0	20d
infra	elasticsearch-master-0	1/1	Running	0	20d
infra	kafkacat-7845b87866-gwndj	1/1	Running	8 (137m ago)	8d
infra	voltha-infra-etcd-0	1/1	Running	0	20d
infra	voltha-infra-fluentd-elasticsearch-fjhpp	1/1	Running	0	20d
infra	voltha-infra-freeradius-859fb6c4c9-ss5mb	1/1	Running	0	20d
infra	voltha-infra-kafka-0	1/1	Running	0	20d
infra	voltha-infra-kibana-668df56fdf-nmslz	1/1	Running	0	20d
infra	voltha-infra-onos-classic-0	1/1	Running	0	20d
infra	voltha-infra-onos-classic-onos-config-loader-7b578bfcb4-b7g6m	1/1	Running	0	20d
infra	voltha-infra-voltha-tracing-jaeger-646c659d6d-t5wqc	1/1	Running	0	20d
infra	voltha-infra-zookeeper-0	1/1	Running	0	20d
kube-flannel	kube-flannel-ds-j99v8	1/1	Running	6 (108d ago)	219d
kube-system	coredns-6d4b75cb6d-shddk	1/1	Running	4 (111d ago)	219d
kube-system	coredns-6d4b75cb6d-x6n6b	1/1	Running	4 (111d ago)	219d
kube-system	etcd-oran-seba	1/1	Running	4 (111d ago)	219d
kube-system	kube-apiserver-oran-seba	1/1	Running	7 (104d ago)	219d
kube-system	kube-controller-manager-oran-seba	1/1	Running	166 (99d ago)	219d
kube-system	kube-proxy-xlxtp	1/1	Running	4 (111d ago)	219d
kube-system	kube-scheduler-oran-seba	1/1	Running	163 (99d ago)	219d
voltha	bbsim0-69485c9f5f-ql598	1/1	Running	0	20d
voltha	voltha-voltha-adapter-openolt-75d5fc78dc-j5ptt	1/1	Running	0	20d
voltha	voltha-voltha-adapter-openonu-59bd5c7dc7-6sjff	1/1	Running	1 (20d ago)	20d
voltha	voltha-voltha-ofagent-6cc7899f94-8fxk8	1/1	Running	0	20d
voltha	voltha-voltha-rw-core-784dfbc4-4jzd7	1/1	Running	0	20d

Figura 17 – *Cluster kubernetes* do VOLTHA.

infraestrutura implementada.

Uma vez que a topologia da rede ainda não possui os equipamentos físicos para o uso, uma alternativa foi o uso do simulador de banda larga, o BBSim (*BroadBand Simulator*) da ONF que faz parte do projeto VOLTHA. Esse simulador permite emular um dispositivo compatível com OpenOLT, por conseguinte, OLTs, portas PON, ONUs, UNIs (*User Network Interface*) e RGs (*Residential Gateway*). A partir da abstração dos equipamentos é possível simular aplicações de gerenciamento, possibilitando uma execução virtualizada de operação. Entretanto, o simulador apresenta somente o plano de controle da rede, sem o plano de dados, e com isso não abrange o tráfego de pacotes entre dispositivos. Então, a partir da implementação do BBSim, foi desenvolvido um plano de dados para o simulador pela equipe do OpenRAN@Brasil da RNP, o que possibilita uma visão completa do sistema emulado com plano de controle e dados, tal qual em um sistema real.

Com esse fim, a emulação do plano de dados se deu com a implementação de um *switch* virtual no núcleo do BBSim, o OVS (OpenVSwitch). Assim, por meio do OVS é possível mapear os componentes emulados em elementos virtuais, como pontes e portas, bem como em portas Ethernet. Para controlar o ciclo de vida dos dispositivos, o BBSim faz uso de máquina de estados, que a cada operação, desencadeia ações responsáveis por mapear o estado dos dispositivos em configurações nas pontes, portas e VLANs. Tais

configurações permitem o envio de pacotes entre os usuários e os serviços.

A OLT simulada no BBSim possui dois tipos de portas, as portas PON e a porta NNI (*Network-Network Interface*), responsáveis por se conectarem às ONUs e à rede, respectivamente. Além disso, as ONUs também possuem dois tipos de portas, as portas UNI, e a porta de comunicação com a OLT. Então, no OVS, as interfaces Ethernet de entrada são adicionadas as portas UNIs, enquanto que a interface Ethernet de saída é adicionada a porta NNI. O circuito fim a fim é implementado emulando as associações entre as portas UNI, PON e NNI, representado na Figura 18. Dessa forma, o tráfego recebido via cliente percorre o circuito ONU-OLT e alcança a rede externa.

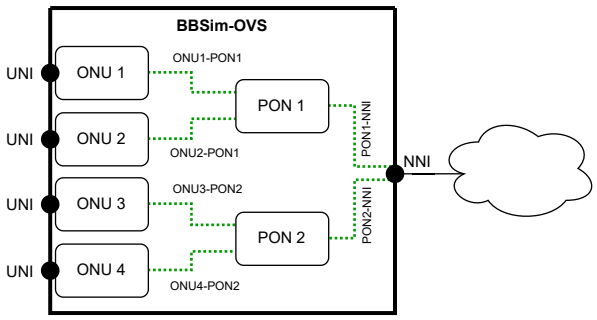


Figura 18 – Mapeamento de portas do BBSim.

Para que o plano de dados sempre reflita o *status* atual do plano de controle, a máquina de estado implementada no BBSim tem papel fundamental. Após qualquer alteração em seu estado, ela desencadeia funções para analisar o ambiente. Em caso de alteração, as configurações são refletidas no OVS e, consequentemente, alteradas no ambiente de experimentação.

O procedimento para implementação da infraestrutura emulada do sistema de rede FTTx é descrito nos passos a seguir:

1. Acesso à CLI do VOLTHA (voltctl);
2. Verificação dos adaptadores OpenOLT e OpenOnu conectados aos equipamentos;
3. Criação da topologia virtualizada com os parâmetros da OLT;

4. Ativação da OLT e das ONUs;

5. Listagem e verificação da topologia empregada e conexão dos equipamentos.

A lista de dispositivos da topologia virtualizada é mostrada na Figura 19, onde se visualiza as informações como identificação dos equipamentos, tipo, *serial number*, *status* administrativo e de conexão.

ID	TYPE	ROOT	PARENTID	SERIALNUMBER	ADMINSTATE	OPERSTATUS	CONNECTSTATUS	REASON
cb1971bf-f872-4003-8503-4fe741c3756e	openolt	true	7243964a-157e-4213-94c3-cbf8c6ced4b9	BBSIM_OLT_10	ENABLED	ACTIVE	REACHABLE	
bcecce86-d008-4723-ab22-6b57b8418ad2	brcm_openomci_onu	false	cb1971bf-f872-4003-8503-4fe741c3756e	BBSM000a0102	ENABLED	ACTIVE	REACHABLE	omci-flows-pushed
ffdc194f-6f67-452a-8e0d-77639324a633	brcm_openomci_onu	false	cb1971bf-f872-4003-8503-4fe741c3756e	BBSM000a0002	ENABLED	ACTIVE	REACHABLE	omci-flows-pushed
f6ba0c14-353f-4fa8-911b-7efdd2631e75	brcm_openomci_onu	false	cb1971bf-f872-4003-8503-4fe741c3756e	BBSM000a0001	ENABLED	ACTIVE	REACHABLE	omci-flows-pushed
0edd801e-95e7-4eae-833f-bc68130771a9	brcm_openomci_onu	false	cb1971bf-f872-4003-8503-4fe741c3756e	BBSM000a0101	ENABLED	ACTIVE	REACHABLE	omci-flows-pushed

Figura 19 – Topologia virtualizada da rede FTTx.

Na Figura 20 é mostrada a topologia da rede PON criada no simulador com uma arquitetura de 4 ONUs e 1 OLT. Essa arquitetura é criada com duas portas PON provenientes da OLT que se conectam aos divisores de sinais alcançando cada porta de interface das ONUs, em que, cada ONU possui uma porta UNI na interface de usuário totalizando 4 portas UNI para operação. No outro lado da OLT está uma porta NNI para conexão externa da rede e a porta de gerenciamento para conexão com adaptadores de OLT virtualizados.

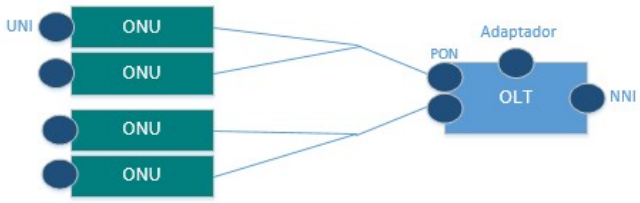


Figura 20 – Arquitetura representativa da topologia simulada de uma rede FTTx com 4 ONUs e uma OLT.

Após a implementação dos equipamentos na estrutura do VOLTHA, a visualização é feita a partir da camada do controlador. A topologia é identificada pelo controlador ONOS e assume-se o estado de autenticação para a operação dos equipamentos. A partir da API "*aaa*", é possível obter as portas que estão autenticadas e disponíveis para a ativação de serviços na camada óptica. O número de portas de autenticação não necessariamente é igual ao número de equipamentos da topologia, sendo que, alguns podem estar desativados, ocupados ou indisponíveis, somente identificando

os usuários autenticados para operação. Na Figura 21, observam-se os pontos de terminais autenticados pelo controlador e disponíveis para a ativação de clientes.

```
karaf@root > aaa-users
of:00000a0a0a0a0a0a/256: AUTHORIZED_STATE, last-changed=8d2h ago, mac=2E:0A:00:01:00:00, subid=BBSM000a0001-1, username=user
of:00000a0a0a0a0a0a/512: AUTHORIZED_STATE, last-changed=8d1h ago, mac=2E:0A:00:02:00:00, subid=BBSM000a0002-1, username=user
of:00000a0a0a0a0a0a/65792: AUTHORIZED_STATE, last-changed=7d3h ago, mac=2E:0A:01:01:00:00, subid=BBSM000a0101-1, username=user
of:00000a0a0a0a0a0a/66048: AUTHORIZED_STATE, last-changed=20d8h ago, mac=2E:0A:01:02:00:00, subid=BBSM000a0102-1, username=user
karaf@root >
```

Figura 21 – Pontos de terminais autenticados para operação de serviços.

Diante das etapas de mecanismos de gerência, realiza-se as configurações de serviço para atribuição aos clientes. Desse modo, a API "*network*" do sistema do controlador é responsável por alterar e aplicar as configurações de serviço para a infraestrutura implementada do FTTx. Entre os principais parâmetros estão: o nome do serviço, a tag do cliente, a tag do serviço, o nome da ONU, endereço MAC dos equipamentos, o perfil de tecnologia, os perfis de largura de banda de *upstream* e *dowstream*, dentre outros. A aplicação das configurações é feita via REST e utilizando um arquivo contendo os dados necessários do tipo dicionário com formato *.json*, que são enviados e armazenados no banco de dados do controlador.

A partir das configurações fornecidas pelo operador, o sistema de gerenciamento apresenta o estado pré-aprovisionamento de um cliente. Através da API "*olt*", pode-se escolher o ponto de terminal a ser aplicado o aprovisionamento de um determinado cliente de acordo com as configurações desejadas para o serviço. Com o comando REST recebido, o fluxo do aprovisionamento de um cliente é realizado, tornando o serviço habilitado para o ponto de terminal selecionado. O *status* da conexão passa de autenticado para ativado, permanecendo até que uma alteração seja realizada por parte do operador. Alterações das configurações e remoção do aprovisionamento podem ser aplicadas em um ou vários clientes sem comprometer a operação do sistema.

No plano de controle, a configuração da camada de serviço e o aprovisionamento de um usuário para conexão do acesso aos serviços digitais são as etapas sistêmicas do gerenciamento da rede SD-PON. Adicionar um assinante, aplicar as configurações de operação e ativar as requisições do sistema foram realizados seguindo um fluxo de im-

plementação de assinante em FTTx.

A partir da ativação do serviço e aprovisionamento do assinante, os dados são mostrados na Figura 22. O controlador armazena um perfil de assinante, que contém informações a respeito dos parâmetros de serviço. Dentre os principais parâmetros presentes estão: o ponto de terminal do usuário, que identifica a porta de conexão da ONU com o endereço da OLT; os valores das tags de cliente e do serviço na conexão com as portas PON; o identificador do perfil de tecnologia, que determina as condições de operação das interfaces das ONUs; o perfil de largura de banda de *uplink* e *downlink* do plano de gerência e do plano de dados do assinante, constituído pelos valores dos parâmetros de taxa de informação e o nome do serviço em operação.


```
{
  "entries": [
    {
      "location": "of:00000a0a0a0a0a0a/512",
      "tagInfo": {
        "uniTagMatch": 0,
        "ponCTag": 10,
        "ponSTag": 15,
        "usPonCTagPriority": -1,
        "usPonSTagPriority": -1,
        "dsPonCTagPriority": -1,
        "dsPonSTagPriority": -1,
        "technologyProfileId": 64,
        "upstreamBandwidthProfile": "User_Bandwidth1",
        "downstreamBandwidthProfile": "User_Bandwidth2",
        "upstreamOltBandwidthProfile": "User_Bandwidth1",
        "downstreamOltBandwidthProfile": "User_Bandwidth2"
        "serviceName": "hsia",
        "enableMacLearning": false,
        "configuredMacAddress": "",
        "isDhcpRequired": true,
        "isIgmpRequired": false,
        "isPppoeRequired": false
      }
    }
  ]
}
```


Figura 22 – Usuário provisionado.

Dado o processo de aprovisionamento do assinante, o fluxo direcionado para a execução da camada de gerência do sistema é mostrado na Figura 23, sendo representado (a) a tela de saída da CLI do ONOS e (b) o fluxograma que descreve de forma visual os passos retratados em (a). Então, para a implementação do fluxo de serviço, o controlador inicia identificando o ponto de terminal do assinante. Diante disso, algumas configurações são removidas da ONU, como a VLAN padronizada do cliente e são adicionadas novas configurações dos protocolos DHCP (*Dynamic Host Configuration Protocol*) e EAPOL (*Extensible Authentication Protocol over LAN*), seguindo para a aplicação do plano de controle com os parâmetros do serviço. Para completar o processo, os perfis da largura de banda para o plano de dados são aplicados, provendo ao assinante o serviço orquestrado.

Enquanto isso, os parâmetros recebidos são utilizados pelo OVS, no plano

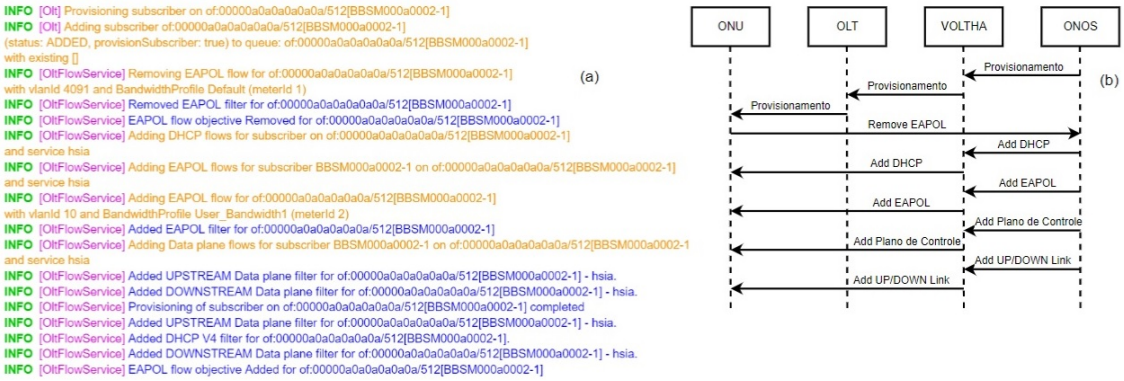
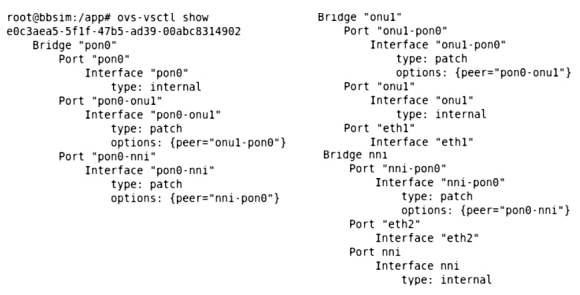
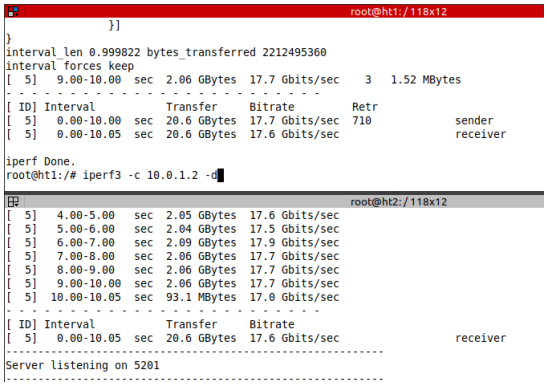


Figura 23 – Fluxo de aprovisionamento da camada de controle.

de dados, para configurar as portas, tipo e capacidade de tráfego entre os enlaces. A complexidade das configurações e operações são abstraídas e mostradas na Figura 24a. As configurações geradas a partir da interação entre controlador de rede definida por software (SDN), VOLTHA, máquina de estados do BBSim e OVS proporcionam a transferência de dados entre os dispositivos, como visto na Figura 24b



(a) Mapeamento de portas pelo OVS.



(b) Transferência de dados na rede.

Figura 24 – Esquema de representação do plano de dados com OVS.

8 Domínio DWDM

Este capítulo apresenta na seção 8.1, uma descrição da arquitetura e elementos para implantação do domínio DWDM, incluindo as definições e detalhamento da topologia e hardwares utilizados no ambiente de testes pré-testbed. São apresentados os módulos e softwares instalados para o seu correto funcionamento no ambiente e os testes realizados no período pré-testbed.

8.1 "Pré-testbed" - Simulações

8.1.1 Descrição da arquitetura de controle

Para implementação do testbed, o ODTN (*Open and Disaggregated Transport Network*), projeto desenvolvido pela ONF, foi utilizado como plataforma fundamental. Este projeto habilita a interconexão de redes de transporte ópticas, utilizando equipamentos com padrões comuns e software de código aberto, permitindo o uso de transponders de diferentes fabricantes. No plano de controle utiliza o controlador ONOS, que descobre de forma automática e transparente os elementos da rede óptica de transporte, além de possibilitar o controle da mesma. E no plano de dados, utiliza equipamentos "white box" com sistema operacional OcNOS, que permite a integração com o ONOS no plano de controle.

8.1.2 Descrição da Topologia implementada

Para viabilizar a implementação do cenário "pré-testbed", incluindo testes preliminares no ambiente, a topologia da figura 25 a seguir foi utilizada.

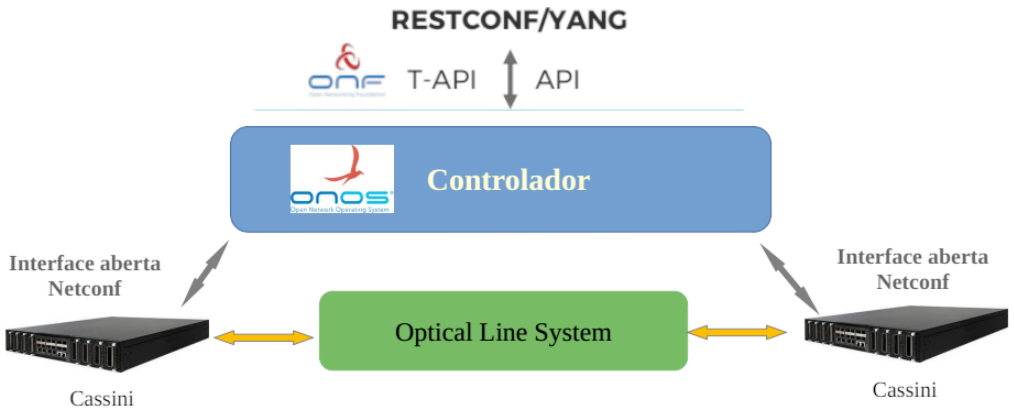


Figura 25 – Topologia testbed DWDM

Nesta topologia temos dois equipamentos DWDM conectados por um enlace de fibra óptica, e controlados por um controlador ONOS, executado como container em um servidor externo. Os equipamentos físicos possuem sistema operacional OcNOS, que utiliza o protocolo Netconf para interação com a interface southbound do controlador ONOS. Este controlador possui driver customizado para acesso, leitura e modificação dos parâmetros dos recursos de hardware. Detalhes serão apresentados nas próximas seções deste capítulo.

8.1.3 Características do Hardware utilizado

O hardware utilizado é o transponder Cassini da Edgecore. Esse transponder é modular e oferece uma plataforma aberta, suportando soluções desagregadas de software, para utilização em transporte óptico para interconexão de data centers, re-

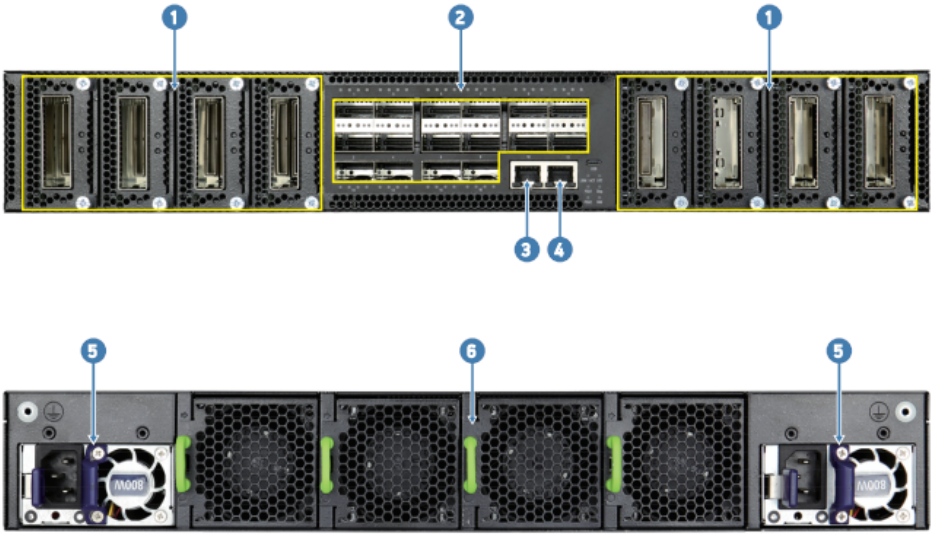
des metropolitanas e redes de acesso. Este equipamento pode funcionar com o sistema operacional OcNOS, e possibilita o uso do protocolo Netconf como interface, o que simplifica o trabalho de integração entre o sistema operacional de rede e o hardware óptico subjacente.

Os modelos instalados no testbed são Cassinis AS7716-24SC, que operam nas camadas L1, L2 e L3, com 16 portas de 100 Gb QSFP28, 8 line cards slots, oferecendo vazão de 3,2Tbps.

A seguir, as principais características do transponder Cassini são destacadas:

- Alta capacidade de vazão oferecendo até 3,2 Tbps.
- Possibilidade de transmissão com placa plug-in coerente 200G 16QAM.
- Funções de switch e transporte na mesma caixa.
- Aberto, flexível e facilidade de uso com biblioteca de transporte que oferece suporte às funcionalidades de APIs e também monitoramento automatizado, relatórios de falha, entre outras ferramentas, abstraindo a complexidade da transmissão L1.
- Alta confiabilidade com ventilador redundante de 60 mm (conjuntos de ventiladores de rotor duplo 3+1) e alimentação redundante (1+1 PSU).
- Comutador de hardware pré-carregado com Open Network Install Environment (ONIE) para carregamento automatizado de NOS de código aberto e outros comerciais compatíveis.

A figura 26 a seguir apresenta as principais características físicas do transponder Cassini AS7716-24SC utilizado no ambiente pré-testbed



Description			
1.	8 x DCO cards	4.	Console port
2.	16 x QSFP28 100G Ethernet ports	5.	PSUs
3.	Management port	6.	Hot-swappable 3 + 1 redundant fans

Figura 26 – Características dos Cassinis

8.1.4 Características da(s) VM(s) utilizada(s)

Para o ambiente ODTN, foi utilizada a implantação do ONOS Classic, juntamente com os demais projetos de cada domínio, e foi executada em uma Máquina Virtual (VM), contendo as seguintes características:

- Processador com 8 núcleos.
- 23.5 GB de memória RAM.
- Disco rígido com 52 GB de capacidade.

8.1.5 Sistemas e Módulos de Software instalados

No projeto ODTN, as versões bases do ONOS já possuem as aplicações necessárias para o funcionamento dos dispositivos reais ou emulados, entretanto é importante citar que existem aplicações fundamentais para as funções de controle e gerenciamento dos dispositivos do domínio DWDM. Cada uma dessas aplicações desempenha um papel específico no contexto de uma rede óptica desagregada, fornecendo funcionalidades relacionadas ao reconhecimento e funcionamento dos transponders Cassinis, bem como, na configuração desses dispositivos do controlador. A utilização dessas aplicações contribui para a operação eficiente e escalável de uma rede DWDM baseada no projeto ODTN. Essas aplicações podem ser vistas com detalhes na seção 9.1.2.1.3 do capítulo 8 deste relatório.

Este ambiente foi implementado em containers Docker para os testes "pré-testbed".

Nos equipamentos ópticos o sistema operacional OcNOS versão EC AS7716-24SC-OcNOS-5.1.201-OTN MPLS-S0-P0, da empresa IPInfusion é utilizado. O sistema OcNOS é um NOS modular, multitarefa e programável, que pode ser executado em todo o portfólio de plataformas Open Compute, por meio de uma única imagem, e se mostrou adequado até o momento, ao longo dos testes realizados e descritos a seguir. Importante mencionar que durante o período dos testes foi verificada a possibilidade de alternativas ao OcNOS, inclusive open source, como o Stratum. Entretanto não existe alternativa viável atualmente. Por exemplo, no nosso ambiente temos instalados tranceivers ópticos da Fujitsu, e este fabricante não oferece os drivers necessários para integração com o sistema aberto Stratum.

8.1.6 Sistemas e Módulos de Software desenvolvidos/customizados

Além das aplicações multidomínio instaladas no controlador, para a correta identificação e funcionamento com os dispositivos transponders cassinis com o sistema operacional OcNOS v5.1 citado na seção anterior, foi necessária a instalação de um

driver chamado cassini-OcNOS5. Este driver foi desenvolvido recentemente pela equipe de desenvolvimento da ONF, e até o momento da escrita deste relatório, ainda não foi incluído em uma das versões de lançamento do ONOS.

Para mais detalhes, ver seção 8.1.1.1 do capítulo 8.

8.1.7 Testes realizados no Pré-testbed

Foram executados três cenários de testes, descritos a seguir:

Cenário 1: Topologia de transporte óptico DWDM formado por dois transponders ópticos Cassini e um enlace de fibra óptica entre eles para testes de funcionalidades do sistema operacional OcNOS. A figura 27 a seguir apresenta o cenário de teste.

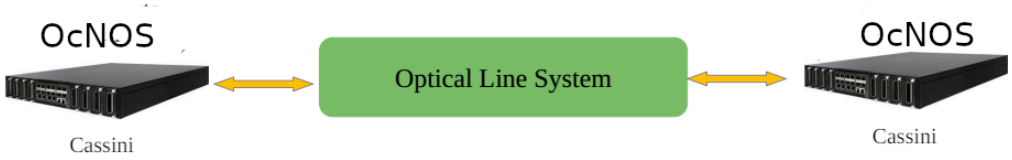


Figura 27 – Cenário 01

Neste cenário foram testados especificamente o estabelecimento de circuitos ópticos entre os dois transponders e a variação de parâmetros dos circuitos ópticos como tipo de modulação, frequência, potência e velocidades de transmissão. A figura 28 a seguir apresenta um exemplo de uma das configurações realizadas com sucesso no sistema OcNOS para modificação de um parâmetro óptico, neste caso específico o tipo de modulação de uma interface.

Modificação de dp-16qam para dp-qpsk
Cassini1#
Após a troca o estado vai para *down*
Até a configuração do mesmo tipo de modulação no outro Cassini, onde o estado volta a *up*

```
Cassini01(config-netif)#modulation-format dp-qpsk
Cassini01(config-netif)#commit
2022 Nov 17 11:36:56.678 : Cassini01 : NSM : CRITI : [IFMGR_IF_DOWN_2]: Interface ce24 changed state to down
2022 Nov 17 11:36:56.696 : Cassini01 : NSM : CRITI : [IFMGR_IF_DOWN_2]: Interface ce23 changed state to down
Cassini01(config-netif)#
Cassini01(config-netif)#
Cassini01(config-netif)#
Cassini01(config-netif)#
Cassini01(config-netif)#2022 Nov 17 11:37:23.311 : Cassini01 : CMM : CRITI : [CMM_TAI_2]: Current Module General Status for Slot: 4 not OK
Module General Status : RX-Network-Loss-Of-Lock

2022 Nov 17 11:37:23.311 : Cassini01 : CMM : CRITI : [CMM_TAI_2]: Client Egress-PCS  Alarms Detected for Slot: 4 Hostif: 0
Egress-PCS  Current Alarms : Remote-Fault

2022 Nov 17 11:37:23.311 : Cassini01 : CMM : CRITI : [CMM_TAI_2]: Client Ingress-PCS Alarms Detected for Slot: 4 Hostif: 0
Ingress-PCS Current Alarms : Loss-Of-Block-Lock
                             Loss-Of-Alignment-Marker-Lock
                             Loss-Of-Alignment

2022 Nov 17 11:38:12.311 : Cassini01 : CMM : CRITI : [CMM_TAI_2]: Current Module General Status for Slot: 4 not OK
Module General Status : RX-Loss-Of-Signal
                        RX-Network-Loss-Of-Lock

2022 Nov 17 11:38:12.311 : Cassini01 : CMM : CRITI : [CMM_TAI_RX_LOS_2]: RX-LOS alarm Detected for Slot:4 Netif:0

2022 Nov 17 11:38:12.311 : Cassini01 : CMM : CRITI : [CMM_TAI_2]: Netif Input-Power Low-Alarm Detected for Slot:4 Netif:0
2022 Nov 17 11:38:35.037 : Cassini01 : NSM : CRITI : [IFMGR_IF_UP_2]: Interface ce23 changed state to up
2022 Nov 17 11:38:37.445 : Cassini01 : CMM : CRITI : [CMM_TAI_2]: Client Ingress-PCS Alarms Detected for Slot: 4 Hostif: 0
Ingress-PCS Current Alarms : BIP-Error
```

Estado do Cassini1 com a nova modulação

```
Cassini01#sh coherent-module summary
-----------------------------------------------
Slot License Information
-----------------------------------------------
Maximum Licenses     : 2
Available Licenses   : 0
Used Licenses        : 2 [Slots : 1, 4]

-------------------------------------------------------------------------------
Slot ModuleStatus FAWS  NetifOperStatus Modulation/OperationalMode InputPower  preFECBER |
                                                                               LaserFreq

-------------------------------------------------------------------------------------------
4    Ready        CLEAR ready          dp-qpsk                    -1.35dBm    0.000000e+00
                                                                               193500000000000Hz
```

Estado do Cassini2 com a nova modulação

```
Cassini02#sh coherent-module summary
-----------------------------------------------
Slot License Information
-----------------------------------------------
Maximum Licenses     : 2
Available Licenses   : 1
Used Licenses        : 1 [Slots : 4]

-------------------------------------------------------------------------------
Slot ModuleStatus FAWS  NetifOperStatus Modulation/OperationalMode InputPower  preFECBER |
                                                                               LaserFreq

-------------------------------------------------------------------------------------------
4    Ready        CLEAR ready          dp-qpsk                    -1.07dBm    8.230719e-10
                                                                               193500000000000Hz
```

Figura 28 – Configuração de modulação no OcNOS

Cenário 2: Expansão do cenário 1 com a inclusão de terminais emulados na topologia para testes de injeção de tráfego nos Cassinis e validação do transporte do tráfego entre eles.

A figura 29 a seguir apresenta a topologia implementada com os Cassinis e o enlace de fibra conectando os mesmos, conforme cenário anterior, e a inclusão de dois terminais virtuais em uma máquina DTN (*Data Transfer Node*). Todos equipamentos são interligados por um switch Stordis.

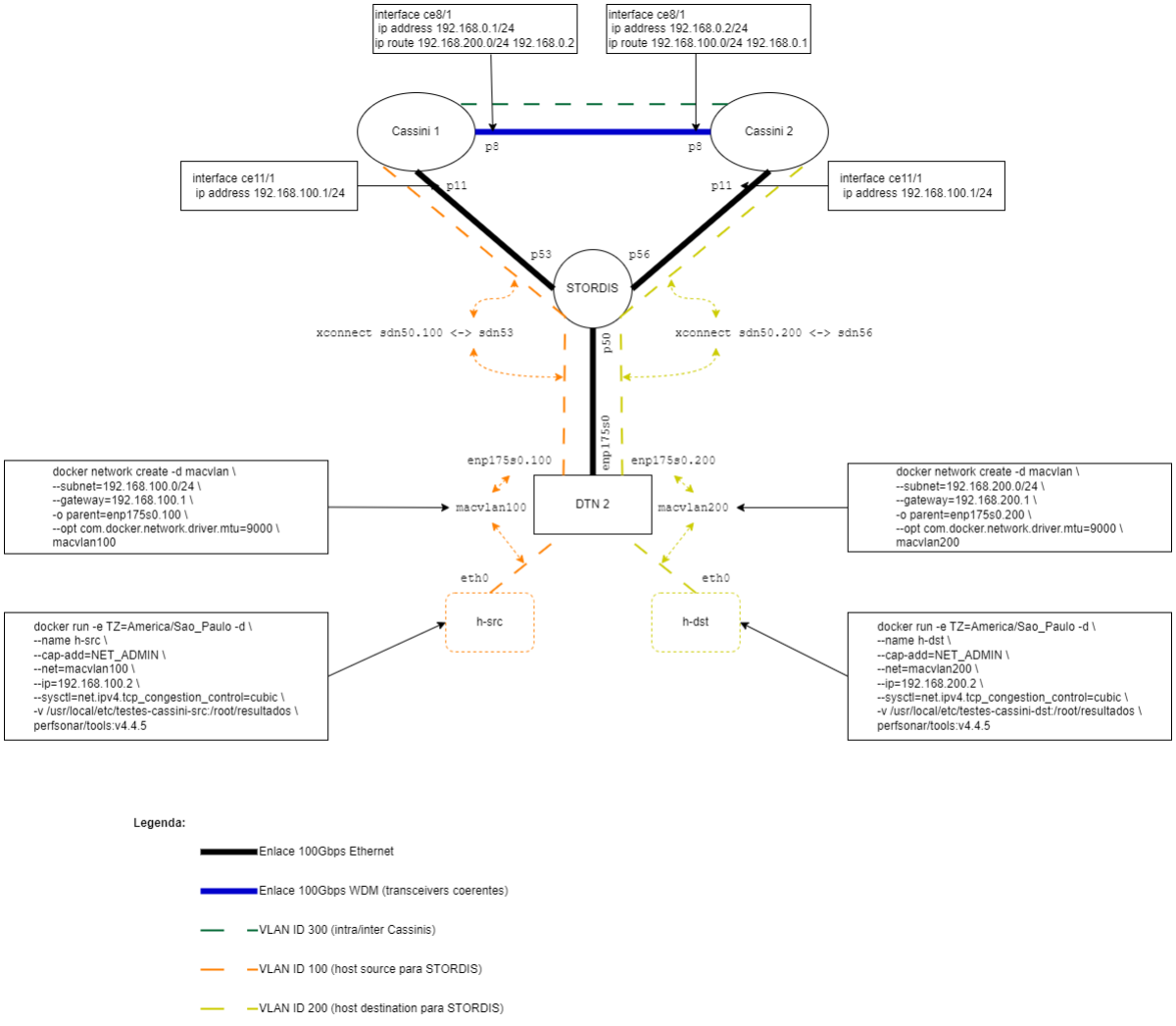


Figura 29 – Topologia:Cassinis, Stordis e DTN

Nos testes de tráfego os dois transponders estavam conectados a um mesmo DTN, em vez de duas máquinas distintas, pois no momento da realização dos testes

apenas uma máquina estava disponível. Para a realização dos testes foram utilizados dois containers Docker e uma rede virtual, conectando os mesmos e os Cassinis por meio de um switch Stordis, com a finalidade de se obter as redes internas e gerar o tráfego entre ambos.

O software para o geração do tráfego entre os transponders foi o iPerf, versão 3.0. O objetivo principal foi a verificação das capacidades de transmissão dos Cassinis. Os resultados obtidos nos testes foram por volta de 20 Gbps, com o uso do protocolo de transporte UDP. As capacidades nominais dos equipamentos chegam a 100Gbps e podemos supor que pela topologia criada e pelo ambiente de rede desenvolvido, as taxas ficaram restritas. Teste mais contundentes serão realizados com a topologia final do testbed concluído.

Outro teste foi relacionado com manipulação dos parâmetros de modulação com tráfego existente entre os equipamentos, diferente do cenário anterior onde apenas o teste do OcNOS e a sincronização entre os Cassinis era o objetivo. Este parâmetro foi alterado para percepção de como o canal se comportaria e não houve alterações nas taxas de download. Os resultados foram apresentados em workshop interno.

Cenário 3: Integração dos Cassinis com o controlador ONOS para criação de um ambiente ODTN (*Optical Disgregated Transport Network*).

Neste cenário foram testadas funcionalidades como descoberta de recursos, configuração, provisionamento e gerenciamento de circuitos ópticos no ambiente ODTN. Com estes testes foi consolidado um entendimento básico sobre o funcionamento do transponder Cassini, e de como o equipamento e suas interfaces funcionam com o protocolo Netconf (*Network Configuration*) em seu sistema operacional OcNOS, para uso do controle dos seus recursos pelo controlador ONOS.

A princípio o ONOS apenas reconhecia os Cassinis e o enlace óptico entre eles, entretanto as interfaces ópticas e elétricas não eram reconhecidas. Após várias interações com a equipe da ONF esse problema foi resolvido, por meio da identificação da falha do

driver utilizado pelo ONOS e a adoção do novo driver customizado para o projeto. Mais detalhes sobre o ONOS e o driver podem ser vistos na seção 9.1.2.2.

A figura 30 a seguir apresenta o ONOS e a topologia dos Cassinis descobertos por ele por meio da interface gráfica dos sistema.

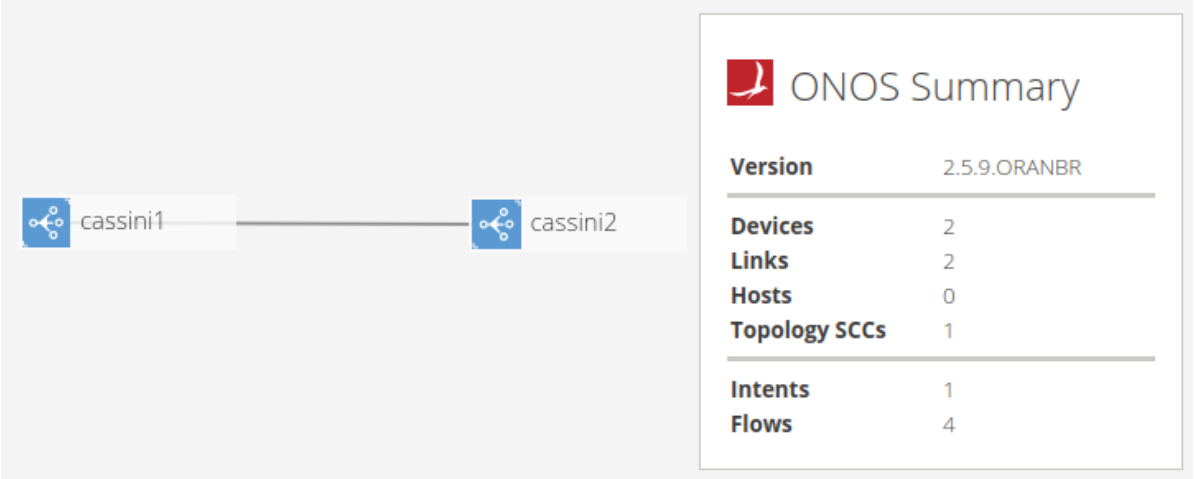


Figura 30 – Visualização dos Cassinis na GUI do ONOS

Esta etapa de testes será aprofundada e terá continuidade posteriormente no ambiente do testbed em funcionamento.

9 Controlador ONOS Multidomínio

O controlador ONOS (*Open Network Operating System*) é um projeto de código aberto, desenhado como controlador distribuído para as arquiteturas SDN com o principal objetivo de resolver os problemas de escalabilidade, disponibilidade e desempenho das redes modernas. Visto isso, ele possui uma extensível e programável plataforma para fornecer flexibilidade na criação de serviços dinâmicos de rede com interfaces programáticas simplificadas e um conjunto de aplicações disponíveis para uso.

Essas vantagens só são possíveis em decorrência da sua arquitetura que é dividida em Núcleo Distribuído e Interfaces NB (*Northbound*) e SB (*Southbound*). O Núcleo distribuído possibilita alta disponibilidade e escalabilidade pois foi projetado para ser compartilhado em vários nós de um cluster, ele é o componente central do ONOS, responsável pela coordenação e tomada de decisões. Quanto a suas interfaces, a *Northbound* realiza a comunicação com os aplicativos ONOS, atuando como uma simplificação de nível amplo da rede o que facilita a sua interação com o controlador. Já a *Southbound* age como facilitador de comunicação entre o controlador e os dispositivos de via protocolos, como OpenFlow e P4Runtime.

Além disso, outra vantagem da sua arquitetura é a possibilidade da utilização do Atomix [4], um *framework* reativo em Java e Golang, projetado para sistemas distribuídos resilientes a falhas, garantindo a sua utilização e integração em forma de cluster. A ONF (*Open Network Foundation*) apresenta algumas outras vantagens devido à arquitetura do controlador, como:

- Alta disponibilidade por meio de *clustering* e gerenciamento de estado

distribuído;

- Escalabilidade por meio de *clustering*;
- Abstração por meio das interfaces *Northbound* (NB) para visualização da rede de forma global;
- Interface *Southbound* (SB) plugável para suporte OpenFlow, P4Runtime e protocolos novos ou legados;
- Interface gráfica do usuário;
- API REST para acesso a abstrações NB e *Command-line Interface* (CLI);
- Suporte para configuração de fluxo proativo e reativo.

Além de todas essas vantagens, o controlador SDN ONOS é o principal controlador utilizado em iniciativas da ONF com intuito de gerenciar e controlar diferentes domínios tecnológicos, como redes de transporte, redes ópticas de acesso passivo e entre outras. A Figura 31 apresenta algumas iniciativas da ONF, as quais estão sendo aplicadas durante o desenvolvimento do projeto, com utilização de respectivos onos customizados adaptado aos domínios.

Figura 31 – Iniciativas da ONF e seus respectivos ONOS.

9.1 Desenvolvimento do Controlador Multidomínio

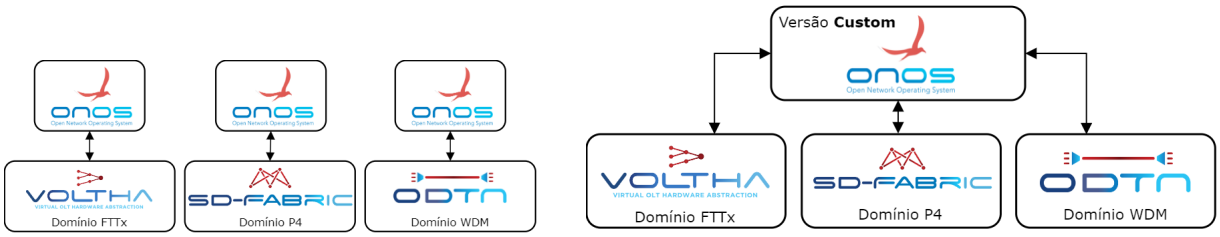
A ONF tem criado e mantido diversas iniciativas que utilizam o ONOS como controlador de redes definidas por *software*, alguns destes projetos utilizam os domínios

investigados neste projeto, como domínio DWDM utilizado no projeto ODTN (*Open and Disaggregated Transport Network*) [5], o qual utiliza esta tecnologia para criar uma arquitetura de rede de transporte óptica desagregada com o uso fundamental de transponders. O domínio FTTx está inserido ao projeto VOLTHA (*Virtual OLT Hardware Abstraction*) [6] que fornece uma abstração de *hardware* virtual para interfaces de OLTs e ONUs, importantes para redes passivas ópticas as quais são gerenciadas pelo controlador ONOS. E por último, o domínio P4 está associado ao SD-Fabric [7], projeto que visa criar uma arquitetura de rede programável otimizada para nuvem de borda e utiliza conceitos de SDN com a presença essencial dos *switches* Stratum, que é um sistema operacional de *switch* independente e de código aberto com suporte ao P4.

Desse modo, o controlador ONOS passa a ter especificidades para funcionar em cada projeto, como: versão do controlador, aplicações e protocolos. Visto isso, o objetivo de criar um ambiente com controle multidomínio passa a ter lacunas a serem preenchidas durante o processo de desenvolvimento e implementação, portanto elaboraram-se duas possíveis soluções de implementação do plano de controle:

- **Solução A**: Utilizar a ferramenta de clusterização do controlador ONOS para conectar diferentes versões do controlador, assim criando um ambiente com três instâncias do controlador as quais os dispositivos heterogêneos seriam gerenciados pelo respectivo controlador do domínio, essa estrutura é apresentada pela Figura 32a;
- **Solução B**: Criar uma versão customizada do controlador reunindo todas as especificidades de cada domínio, com objetivo de gerenciar todos os domínios por meio de um controlador único. A Figura 32b, demonstra a lógica que seria implementada, apesar de estar demonstrado apenas uma instância do controlador customizado, esta estrutura suporta mais de um controlador para atender a necessidade da rede

Visto isso, fez-se um estudo aprofundado sobre as duas soluções de maneira



(a) Solução A. (b) Solução B.

Figura 32 – Possíveis Soluções de implementação do plano de controle.

simultânea. A possibilidade do uso de diferentes versões do controlador em cluster para a gerência do ambiente multidomínio não obteve resultados satisfatórios, os testes visando à integração entre diferentes versões utilizaram como base a característica do Atomix de integrar múltiplas réplicas do controlador ONOS. Desse modo, foram realizadas mudanças no helm *chart* de implementação do controlador com intuito de realizar a conexão entre as diferentes versões do ONOS e o Atomix. Nesta fase de testes, ocorreram erros durante o processo de sincronização entre algumas versões, onde apenas um número reduzido de diferentes versões (ex: 2.5.9 a 2.5.5) se mostraram compatíveis, devido a grande diferença entre os componentes do *core* do ONOS e das suas diversas aplicações.

Visto a impossibilidade de implementação da Solução A e avanços significativos de implementação da Solução B, optou-se pela solução visando à utilização de um controlador único capaz de gerenciar os projetos de cada domínio tecnológico e integrar os dispositivos, como os comutadores Stratum, Cassinis, OLTs e ONUs.

Portanto, esta seção apresenta as etapas de desenvolvimento da segunda solução, que inicialmente passa pelo estudo sobre o versionamento do controlador para encontrar a versão base de desenvolvimento. Em seguida, com a definição da versão base de desenvolvimento do controlador, por meio da investigação das aplicações necessárias para o funcionamento de cada domínio. Por último, foi realizada a geração da versão customizada do ONOS, a qual se deu pelo mapeamento das aplicações que tiveram seus códigos fonte obtidos para compilação das aplicações no ONOS multidomínio. Estas etapas, estão descritas nas Subseções 9.1.1 e 9.1.2 com maiores detalhes.

9.1.1 Versionamento

Inicialmente, para a criação do ONOS Multidomínio, foram conduzidos estudos sobre o versionamento do controlador com o objetivo de definir a versão base de desenvolvimento para a versão customizada capaz de gerenciar os diferentes domínios. O objetivo era selecionar a versão mais recente e funcional do controlador que fosse compatível com as últimas versões dos projetos SD-Fabric, VOLTHA e ODTN.

Inicialmente, a investigação para determinar a versão base do ONOS foi realizada por meio de testes utilizando transponders ópticos Cassinis e *switches* Stratum, que foram integrados e emulados pelo emulador óptico CNETLAB [RNP 2023] [8], uma ferramenta para emular dispositivos e topologias de rede em contêineres Docker, do projeto SDN Multicamadas (SDNM) [9] da Rede Nacional de Ensino e Pesquisa (RNP). Além disso, por meio de testes levantados com versões anteriores do projeto VOLTHA, foi possível investigar a compatibilidade dos dispositivos do projeto, emulados pelo BBSim (*Broadband Simulator*) [10]. Dessa forma, por meio da virtualização do controlador SDN responsável pelo gerenciamento da topologia, foi possível verificar o reconhecimento do controlador sobre os dispositivos e enlaces em ambos os domínios tecnológicos por meio de testes com as topologias. Estes testes variaram a topologia para manter os domínios dispostos de maneira isolada e, também, de maneira integrada.

	2.5.0	2.5.1	2.5.2	2.5.3	2.5.4	2.5.5	2.5.6	2.5.7	2.5.8	2.5.9
P4 - Stratum	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔
DWDM - Cassini	✔	✔	✔	✔	✔	✘	✘	✘	✘	✘
FTTx - OLT e ONU	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔

Tabela 3 – Compatibilidade dos dispositivos com versões base do ONOS Classic.

A Tabela 3 demonstra as informações de compatibilidade das versões a partir da versão 2.5.0 com ambos os dispositivos, as quais podem concluir que o limite de funcionamento é a versão 2.5.4 com ambos os dispositivos, isto ocorre pela limitação dos Cassinis que a partir da versão 2.5.5 apresentam erros relacionados ao reconhecimento do controlador sobre estes enlaces, logo, apesar do ODTN não exigir versões específicas do ONOS, o projeto passa a estar limitado a versão 2.5.4. Somado a isso, também

foram investigados a versão do ONOS implementadas nas últimas versões dos projetos VOLTHA (2.12) e SD-Fabric (1.4.0), sendo elas 2.5.9 e 2.5.8, respectivamente.

Com isto, apesar da versão mais recente suportada pelo projeto ODTN ser a versão do ONOS Classic 2.5.4, foi feita uma implementação da versão customizada para a versão 2.5.9, devido ao requisitos dos projetos SD-Fabric e VOLTHA, gerando uma versão customizada do controlador denominada como ORAN-ONOS. Esta implementação, bem como, os detalhes da construção dessa imagem serão detalhados nas seções seguintes.

9.1.2 Imagem ORAN-ONOS

Como citado na subseção 9.1.1, uma série de requisitos de aplicações e de versionamento do controlador é necessária para cada projeto. Nesse sentido, foi desenvolvida uma imagem customizada do controlador ONOS Classic 2.5.9, intitulada ORAN-ONOS [11], para integrar os diferentes domínios tecnológicos, de modo a possibilitar um gerenciamento logicamente centralizado dos domínios FTTx, P4 e DWDM do projeto OpenRAN@Brasil. Nesta, foram incluídas as aplicações multidomínio necessárias para o funcionamento dos domínios do projeto, como também, incluídos módulos e *drivers* para o reconhecimento dos dispositivos utilizados no projeto OpenRAN@Brasil.

9.1.2.1 Aplicações Multidomínio

Os projetos SD-Fabric e VOLTHA são estruturados em clusters Kubernetes (K8s), resultando em uma arquitetura distribuída composta por vários pods. Esses pods representam componentes ou serviços que se comunicam entre si e com o controlador ONOS. Essa comunicação ocorre por meio das aplicações adicionadas à versão customizada do controlador, as quais são específicas para cada projeto e fornecem funcionalidades de controle sobre os dispositivos. Quanto ao projeto ODTN, as versões bases do ONOS já possuem as aplicações necessárias para o funcionamento dos dispositivos reais ou emulados, mas existem aplicações fundamentais para as funções de controle e gerenciamento dos dispositivos do domínio DWDM.

ODTN		VOLTHA		SD-Fabric
ODTN Service	netconf	aaa	mcast	fabric-tna
yang	optical-model	bng	olt	up4
config	restconf	dhcpl2relay	olttopology	trellis-t3
models.tapi		igmpproxy	ppoagent	trellis control / segmentrouting
odtn-api		kafka	sadis	
protocols.restconfserver		mac-learning		

Tabela 4 – Mapeamento das aplicações referente a cada domínio.

Portanto, foi realizado um trabalho de mapeamento das aplicações dos controladores levantados de cada projeto, como mostrado na Tabela 4, a qual demonstra 3 aplicações necessárias para o funcionamento da estrutura do SD-Fabric, 11 aplicações exclusivas do projeto VOLTHA e 101 aplicações importantes para o projeto ODTN. A Figura 33 ilustra na *Graphical User Interface* (GUI) as aplicações necessárias para criar o ambiente multidomínio, com as 25 aplicações da Tabela 4 ativadas e somadas a outras aplicações fundamentais e nativas da versão base do controlador.

9.1.2.1.1 FTTx

Considerando as aplicações relacionadas ao domínio tecnológico FTTX, cada uma dessas aplicações desempenha um papel específico no contexto de uma rede SD-PON, fornecendo funcionalidades relacionadas à autenticação, provisionamento de serviços, gerenciamento de recursos, encaminhamento de pacotes, suporte a multicast e integração com sistemas externos. A utilização dessas aplicações contribui para a operação eficiente e escalável de uma rede PON baseada no projeto VOLTHA. Dessa maneira, as aplicações listadas na Tabela 2(4), têm suas funções descritas abaixo:

- **AAA**: Desempenha o papel de um servidor NAS e oferece a autenticação RADIUS para as portas;
- **BNG**: Serve para gerenciar e controlar o tráfego de banda larga de maneira eficiente;

	Title	App ID
✓	AAA application for CORD	org.opencord.aaa
✓	BNG	org.opencord.bng
✓	Barefoot Drivers	org.onosproject.drivers.barefoot
✓	Basic Optical Drivers	org.onosproject.drivers.optical
✓	Basic Pipelines	org.onosproject.pipelines.basic
✓	CORD Mcast app	org.opencord.mcast
✓	DHCP L2 Relay	org.opencord.dhcpl2relay
✓	Default Drivers	org.onosproject.drivers
✓	Dynamic Configuration	org.onosproject.config
✓	Fabric-TNA Pipeconf	org.stratumproject.fabric-tna
✓	Fault Management	org.onosproject.faultmanagement
✓	General Device Provider	org.onosproject.generaldeviceprovider
✓	Generic NETCONF Drivers	org.onosproject.drivers.netconf
✓	IETF YANG Models	org.onosproject.models.ietf
✓	IGMPProxy application for CORD	org.opencord.igmpproxy
✓	Kafka integration app	org.opencord.kafka
✓	Mac Learner	org.opencord.maclearner
✓	Multicast traffic control	org.onosproject.mcast

(a) GUI ONOS Aplicações Parte 1

	Title	App ID
✓	NETCONF Provider	org.onosproject.netconf
✓	ODTN API & Utilities Application	org.onosproject.odtn-api
✓	ODTN Driver	org.onosproject.drivers.odtn-driver
✓	ODTN Service Application	org.onosproject.odtn-service
✓	ONF Transport API YANG Models	org.onosproject.models.tapi
✓	ONOS GUI2	org.onosproject.gui2
✓	OpenConfig Infinera XT3300 YANG Models	org.onosproject.models.openconfig-infinera
✓	OpenConfig RD v0.3 YANG Models	org.onosproject.models.openconfig-odtn
✓	OpenConfig YANG Models	org.onosproject.models.openconfig
✓	Optical Line Terminal App	org.opencord.olt
✓	Optical Line Terminal Topology App	org.opencord.olttopology
✓	Optical Network Model	org.onosproject.optical-model
✓	P4Runtime Drivers	org.onosproject.drivers.p4runtime
✓	P4Runtime Protocol Subsystem	org.onosproject.protocols.p4runtime
✓	P4Runtime Provider	org.onosproject.p4runtime
✓	PPPoE Agent	org.opencord.pppoeagent
✓	Port Load Balance Service	org.onosproject.portloadbalancer
✓	REST Provider	org.onosproject.restsb

(b) GUI ONOS Aplicações Parte 2

	Title	App ID
✓	RESTCONF Application Module	org.onosproject.restconf
✓	RESTCONF Server Module	org.onosproject.protocols.restconfserver
✓	Route Service Server	org.onosproject.route-service
✓	Stratum Drivers	org.onosproject.drivers.stratum
✓	Subscriber And Device Information App	org.opencord.sadis
✓	Trellis Control App	org.onosproject.segmentrouting
✓	Trellis troubleshooting tools	org.onosproject.t3
✓	UP4	org.omecproject.up4
✓	YANG Compiler and Runtime	org.onosproject.yang
✓	gNMI Drivers	org.onosproject.drivers.gnmi
✓	gNMI Protocol Subsystem	org.onosproject.protocols.gnmi
✓	gNOI Drivers	org.onosproject.drivers.gnoi
✓	gNOI Protocol Subsystem	org.onosproject.protocols.gnoi
✓	gRPC Protocol Subsystem	org.onosproject.protocols.grpc

(c) GUI ONOS Aplicações Parte 3

Figura 33 – Aplicações representadas na GUI do controlador ONOS.

- **DHCP L2 Relay**: É um agente de retransmissão DHCP que faz a retransmissão da Camada 2;
- **IGMPProxy**: Recebe e decodifica as mensagens de relatório igmp dos *hosts*;
- **Kafka:** Envia eventos de aplicativo ONOS para o barramento Kafka;
- **Mac Learning**: Examina pacotes DHCP do tipo DISCOVER ou REQUEST e mantém o endereço MAC temporariamente de acordo com as informações de dispositivo, porta e vlanId obtidas desses pacotes;

- **Mcast**: Fornece o tratamento de tráfego *multicast*;
- **OLT**: É responsável por configurar os fluxos necessários para gerenciar um dispositivo OLT conforme relatado pelo VOLTHA;
- **OLT Topology**: Recupera por meio de REST e CLI, informações sobre as OLTs e seus pares de rede aprendidos por meio de pacotes LLDP;
- **PPPoE Agent**: Conhecido como agente Intermediário PPPoE, ele suporta o método de acesso PPPoE e é uma função colocada no Nó de Acesso para inserir a identificação do *loop* de acesso;
- **Sadis**: Define um serviço opcional que forneça uma ponte à infraestrutura do cliente para consulta a informações e disponibilizá-lo para outros serviços/aplicações dentro do ONOS.

Após a instalação das aplicações no controlador, é possível realizar a conexão com os componentes do projeto VOLTHA. Neste cenário, essas aplicações possibilitam a gerência dos componentes de uma rede SD-PON, como o monitoramento e provisionamento de recursos como a velocidade de conexão dos assinantes, a configuração de políticas de qualidade de serviço (QoS) e a detecção de falhas na infraestrutura. A lógica de conexão que se pretende utilizar no *testbed* para o domínio FTTx, por meio do projeto VOLTHA, está ilustrada na Figura 34.

9.1.2.1.2 P4

Considerando as aplicações relacionadas ao domínio P4, cada uma desempenha um papel específico no contexto de uma rede de pacotes programável, fornecendo recursos avançados para funções de controle de redes 5G, abstrações de componentes da rede possibilitando a sua integração, auxílio na resolução de problemas relacionados ao tráfego e a viabilidade na utilização de arquiteturas tipo *leaf-spine*. A utilização dessas aplicações proporciona um ambiente altamente programável e gerenciável. Dessa

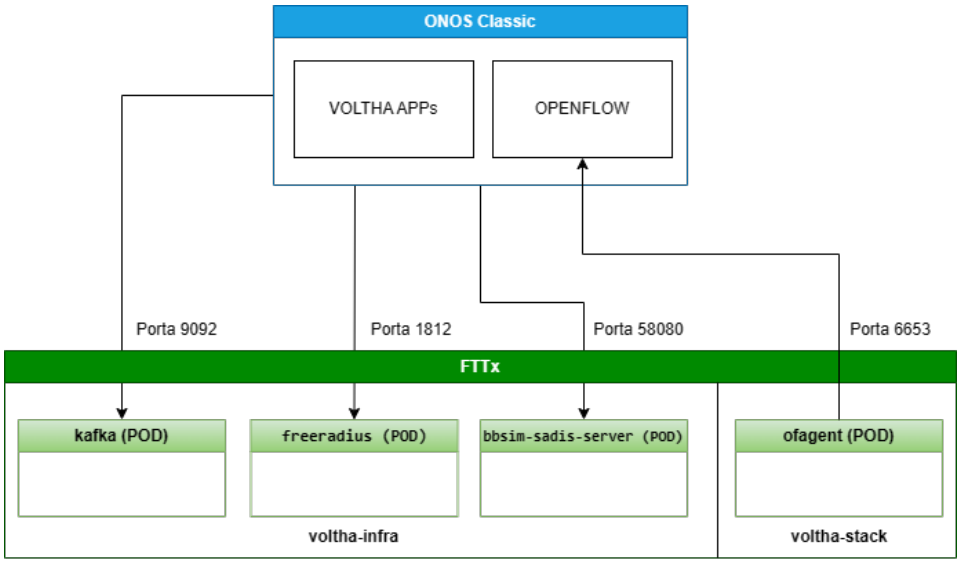


Figura 34 – Diagrama de conexão com o domínio FTTx.

maneira, as aplicações deste domínio listadas na Tabela 4, têm suas funções descritas abaixo:

- **Fabric TNA**: É um programa P4 que realiza o plano de dados do SD-Fabric, uma malha de rede programável habilitada para SDN adaptada para nuvens de borda conectadas ao 5G. Provêm recursos mais avançados para *User Plane Function* (UPF) 4G/5G, *In-band Network Telemetry* (INT), Fatiamento e QoS;
- **UP4**: Abstrai uma rede de um ou mais comutadores Fabric como um "*One-Big-UPF*"virtual, que pode ser integrado a um plano de controle de núcleo móvel 4G/5G;
- **Trellis T3**: É um aplicativo ONOS destinado a ajudá-lo a solucionar problemas relacionados ao tráfego em sua rede, rastreando fluxos e grupos instalados;
- **Trellis Control/Segmentrouting**: É uma aplicação *open-source* de múltiplas propostas para L2/L3 que possibilita a arquitetura *leaf-spine* em*switches* fabric.

Com as aplicações instaladas no controlador, é possível realizar a conexão com os componentes do projeto SD-Fabric, bem como, o dispositivo *switch* Barefoot Tofino. Nesse contexto, estas aplicações possibilitam uma série de recursos com relação a programabilidade dos pacotes e as funções relacionadas ao plano de usuário (UPF), a partir do plano de controle. A lógica de conexão pretendida para o *testbed* com relação ao domínio P4, utilizando o projeto SD-Fabric, é ilustrada na Figura 35.

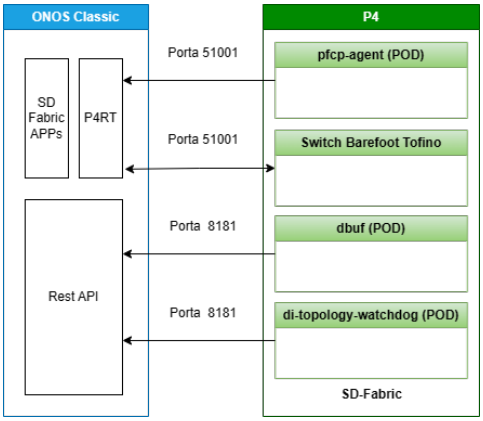


Figura 35 – Diagrama de conexão com o domínio P4.

9.1.2.1.3 DWDM

Considerando as aplicações relacionadas ao domínio DWDM, cada desempenha um papel específico no contexto de uma rede óptica desagregada, fornecendo funcionalidades relacionadas ao reconhecimento e funcionamento dos transponders cassinis, bem como, na configuração desses dispositivos do controlador. A utilização dessas aplicações contribui para a operação eficiente e escalável de uma rede DWDM baseada no projeto ODTN. Dessa maneira, as aplicações deste domínio listadas na Tabela 4, têm suas funções descritas abaixo:

- **ODTN Service**: Aplicação do serviço ODTN que habilita uma série de módulos para funcionamento com os transponders;
- **Yang**: Fornece a capacidade de registrar modelos YANG compilados ou até mesmo compilar arquivos de origem YANG em tempo real.

- **Config**: Fornece meios para rastrear e distribuir dados de configuração de serviço e dispositivo em todo o cluster ONOS. Ela funciona com o tempo de execução YANG para garantir que os dados rastreados sigam os modelos YANG registrados.
- **models.tapi**: Modelos YANG da API de transporte da ONF;
- **ODTN API**: API utilizada por aplicações da ODTN;
- **Restconf**: Módulo de aplicação Restconf;
- **drivers.netconf**: Adiciona suporte para dispositivos usando NETCONF;
- **drivers.odtn-driver**: Drivers relacionados a ODTN;
- **Netconf**: Fornece meios para o ONOS descobrir e acionar o procedimento de *handshake* inicial com NETCONF a partir de informações fornecidas pela configuração de rede.
- **Optical Model**: Modelo de informação óptica do ONOS;
- **protocols.restconfserver**: Módulo de Servidor RESTCONF;

9.1.2.2 Driver cassini-ocnos5

Além das aplicações multidomínio instaladas no controlador, para a correta identificação e funcionamento com os dispositivos transponders cassinis com o sistema operacional (SO) Ocnos v5.1, foi necessária a instalação de um *driver* chamado cassini-ocnos5. Este *driver*, foi desenvolvido recentemente pela equipe de desenvolvimento da ONF [12] e, até o momento da escrita deste relatório, ainda não foi incluído em uma das versões de lançamento do ONOS Classic, como por exemplo, a 2.5.9. Nesse contexto, foi feita a implantação desse *driver* dentro da imagem ORAN-ONOS de forma a possibilitar as operações de reconhecimento das características do dispositivo transponder cassini,

como portas ópticas e de pacotes, bem como o controle efetivo do dispositivo pelo controlador. A Figura 36, ilustra o funcionamento do *driver* dentro do ONOS e como ele realiza a conectividade com o dispositivo óptico.

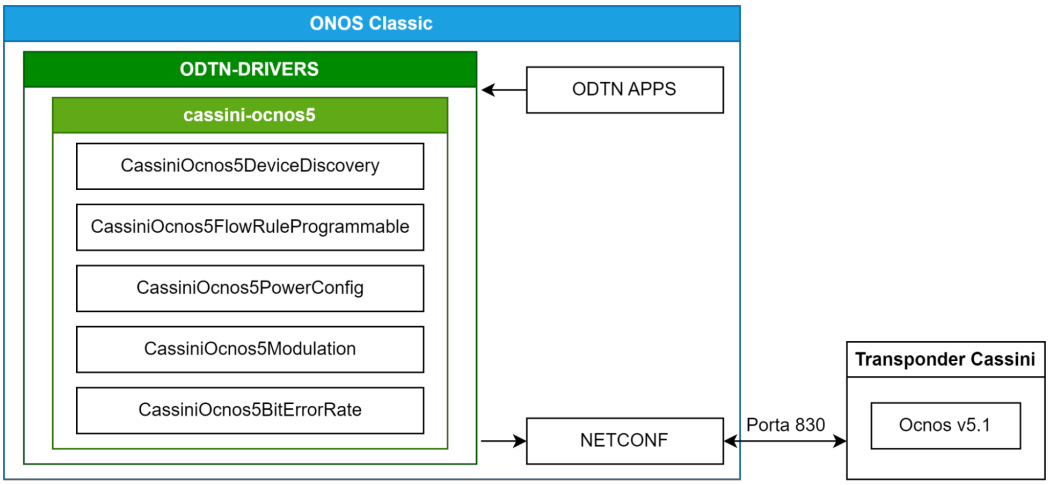


Figura 36 – Diagrama de conexão com o domínio DWDM.

Como mostrado na Figura 36, o *driver* cassini-ocnos5 implementa uma série de módulos que são de extrema importância para o reconhecimento e alteração das funcionalidades do transponder cassini via plano de controle. Nesse sentido, as aplicações do ODTN realizam o papel de habilitar este *driver*, que por sua vez, carrega os módulos necessários para a identificação das funcionalidades do dispositivo óptico via protocolo NETCONF. Além disso, com relação aos módulos, cada um possui uma função específica dentro do controlador, como listado abaixo:

- **CassiniOcnos5DeviceDiscovery**: Implementação da interface de descoberta e descrição do dispositivo;
- **CassiniOcnos5FlowRuleProgrammable**: Implementação da interface programável de regras de fluxo para dispositivo;
- **CassiniOcnos5PowerConfig**: Implementação da interface de configuração energética para dispositivo;

- **CassiniOcnos5Modulation**: Implementação da interface de configuração de modulação para o dispositivo;
- **CassiniOcnos5BitErrorRate**: Implementação da interface de estado da taxa de erro de bit para dispositivo.

9.2 "Pré-testbed" - Simulações

Os testes pré-testbed consistiram na construção de ambientes emulados utilizando ferramentas de virtualização, como o Docker e Kubernetes, para criação de ambientes exclusivos para cada domínio, somado a um ambiente multidomínio, todos gerenciados pelo controlador ORAN-ONOS. O ambiente para testes com apenas dispositivos do DWDM foi construído por meio do emulador óptico CnetLab. Por outro lado, o projeto SD-Fabric foi implementado totalmente em nuvem K8s. Esta estrutura foi a mesma utilizada para implementação dos dispositivos do FTTx por meio do BBSim, além dos demais componentes da estrutura do VOLTHA.

Portanto, as seções seguintes descrevem com mais detalhes as topologias, *softwares*, *hardwares* e testes com as topologias utilizadas na implementação do ONOS, inicialmente, de maneira isolada com cada domínio e, posteriormente, de forma a integrar estes domínios.

9.2.1 Descrição da Topologia implementada

9.2.1.1 FFTx (VOLTHA)

Como supracitado, a infraestrutura e os dispositivos do VOLTHA necessitam do ambiente de nuvem para o seu pleno funcionamento. Dessa maneira, a Figura 37 ilustra a topologia implementada, neste caso de forma a implementar uma OLT e uma ONU. Dessa maneira, primeiro um cluster Kubernetes foi implementado, que é a base para todos os domínios tecnológicos que utilizam o controlador ONOS Classic, após isso, os repositórios atomix, onosproject e opencord foram adicionados no gerenciador

de aplicações do Kubernetes conhecido como Helm, que auxilia na automatização da instalação de dependências no cluster. Por fim, foi instanciado o controlador ONOS junto ao Atomix com todas as aplicações necessárias para o funcionamento do domínio FTTx.

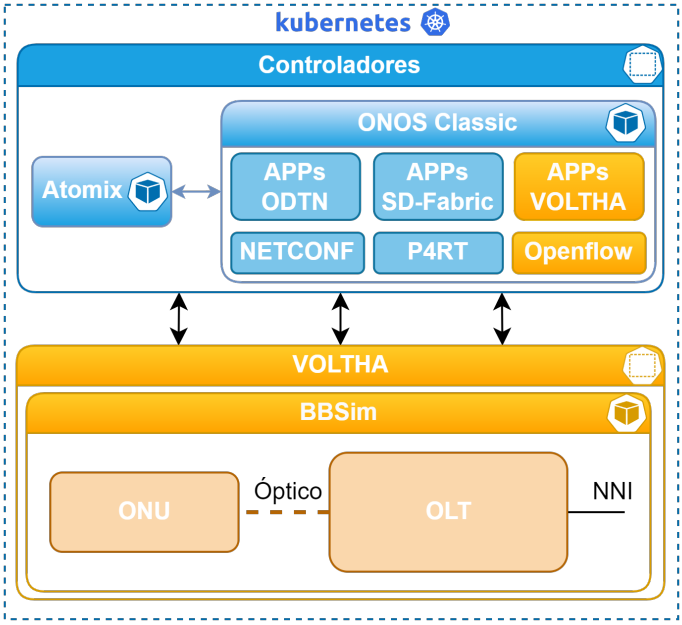


Figura 37 – Topologia implementada para o domínio FTTx.

Após a preparação do ambiente de controle em um *namespace* próprio, a implementação do projeto VOLTHA foi feita primeiro com a instalação das ferramentas de infraestrutura pelo voltha-infra que contém componentes responsáveis por envio de mensagens, barramento de eventos, *logs* e armazenamento de dados. Com a infraestrutura pronta, os componentes principais de funcionamento do projeto foram implementados, incluindo o núcleo que é responsável pelas solicitações e por abstrair os dispositivos em *switches* lógicos. Assim como, os adaptadores para os dispositivos de rede que lidam com as interações específicas dos dispositivos, além do agente responsável pela comunicação entre plano de controle e núcleo, permitindo o gerenciamento das OLTs e ONUs.

Por fim, para testes de funcionamento foi utilizada a ferramenta BBSim (*BroadBand Simulator*), que é uma ferramenta criada para emular um dispositivo compatível

com o adaptador OpenOLT utilizado para conexão entre dispositivos, a exemplo de OLTs (*Optical Line Terminal*) e ONUs (Optical Network Unit). Para os testes foram utilizados os dispositivos ONU e OLT, como também, a interface NNI (Network-to-Network Interface).

9.2.1.2 P4 (SD-Fabric)

O projeto SD-Fabric, assim como o VOLTHA, possui componentes instalados em *cloud*, utilizando um cluster Kubernetes. Nesse sentido, para elaborar uma topologia de rede emulada para este domínio, utilizou-se o exemplo apresentado pelo próprio projeto SD-Fabric [13]. Dessa forma, este exemplo considerou a utilização do *software* de emulação Mininet para emular os dispositivos *switches* Stratum que são utilizados por este domínio no projeto SD-Fabric.

Primeiramente, foi implantado o controlador ONOS Classic no *namespace* de controladores dentro do cluster Kubernetes. Esta implantação, já considera a utilização da imagem do controlador desenvolvida para o OpenRAN@Brasil (ORAN-ONOS), bem como, o carregamento dos módulos e aplicações para o funcionamento multidomínio.

Com o controlador implantado, implementou-se a topologia do plano de dados considerando a emulação de 4 *switches* Stratum BMv2 em uma arquitetura *leaf-spine*, no qual, utilizou-se 2 *switches* do tipo *spine* e 2 *switches* do tipo *leaf*. Além disso, para este exemplo considerou-se a implantação de 6 *hosts*, sendo 4 conectados nas portas do *leaf1* e 2 conectados nas portas do *leaf2*. Todos estes dispositivos foram implantados utilizando o próprio Helm *Chart* disponibilizado pela ONF, no exemplo que considera a completa virtualização do SD-Fabric. Além da topologia emulada utilizando o Mininet, o *Packet Forwarding Control Protocol* (PFCP) *Agent* que está incluso no SD-Fabric Helm *Charts* também foi implantado, de forma a testar a conectividade com o cliente UP4 com o controlador. Com isso, obteve-se a topologia planejada para o domínio P4, utilizando o projeto SD-Fabric, que é ilustrada na Figura 38.

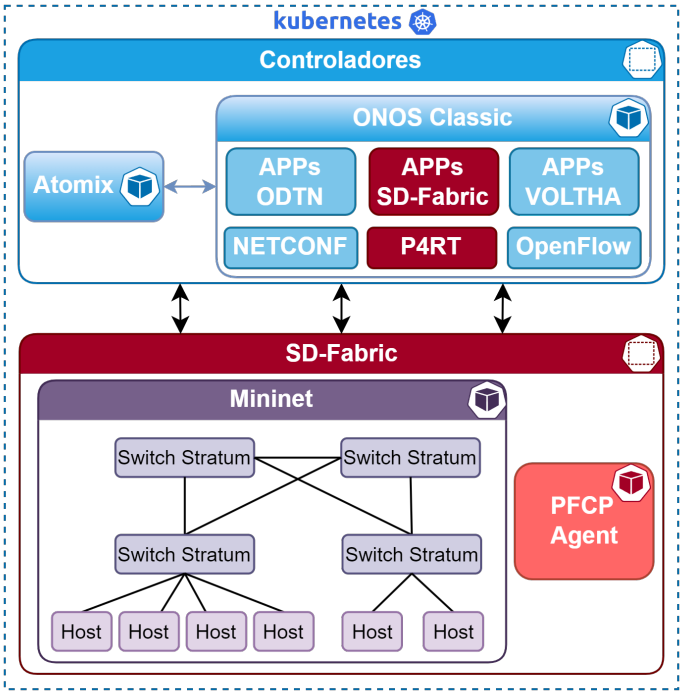


Figura 38 – Topologia implementada para o domínio P4.

9.2.1.3 DWDM (ODTN)

Conforme mencionado anteriormente, o projeto da *Open Networking Foundation* (ONF) relacionado ao domínio DWDM é conhecido como ODTN. Essa iniciativa busca requisitos mais simples de implementação em comparação com outros domínios mencionados anteriormente. Isso se deve ao fato de que sua infraestrutura não requer componentes extras em nuvem, e as versões base do controlador são suficientes para atender às necessidades de aplicações específicas para seu funcionamento.

A topologia implementada, apresentada na Figura 39, envolve o estabelecimento da camada de controle no Kubernetes, onde se encontra o controlador ORAN-ONOS e um Atomix no *namespace* "Controladores". Além disso, através do emulador óptico CnetLab, foram configurados 2 transponders do tipo Cassini, conectados por uma conexão óptica bidirecional. Cada Cassini possui um *host* conectado às suas portas elétricas. Com essa infraestrutura criada pelo simulador, os transponders comunicam-se com o controlador utilizando o protocolo NETCONF, permitindo o uso das funcionalidades

de gerenciamento e monitoramento por meio de suas respectivas aplicações ODTN.

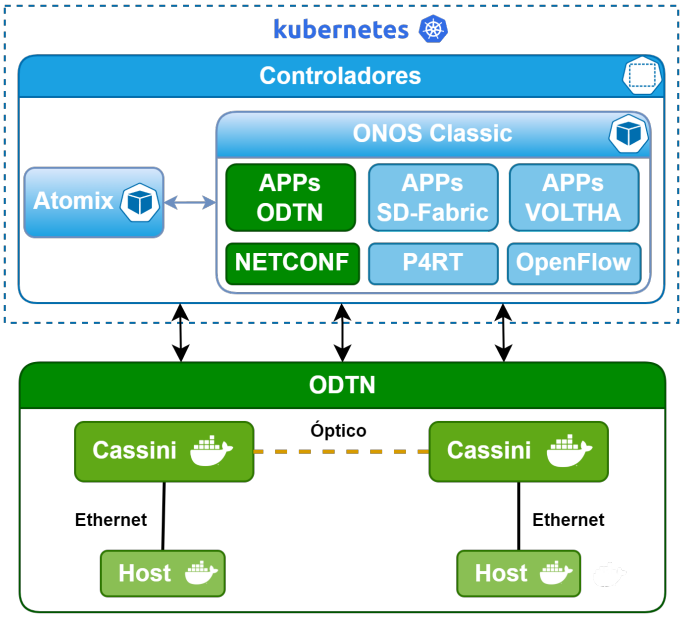


Figura 39 – Topologia implementada para o domínio DWDM.

9.2.1.4 Integração Multidomínio

Com base nas topologias supracitadas, pode-se verificar o funcionamento levando em consideração cada domínio isoladamente. No entanto, de forma a validar a implantação multidomínio do controlador desenvolvido, montou-se uma topologia considerando a conexão simultânea dos 3 domínios do projeto com o mesmo controlador. Nesse sentido, foram implantados o ONOS Classic, VOLTHA e SD-Fabric dentro de um cluster Kubernetes, sendo cada um implementado em um *namespace* diferente do cluster. Com relação ao projeto ODTN, utilizou-se o CnetLab para a emulação dos dispositivos ópticos transponder cassini.

Feita a implantação dos projetos de cada domínio, conectou-se estes ao controlador no Kubernetes, sendo esta conexão feita por arquivos de configuração YAML dentro do próprio cluster ou por meio do envio de informações via API Rest do ONOS Classic com as informações dos dispositivos do plano de dados e dos módulos de cada

domínio para conexão do controlador. A Figura 40 ilustra a topologia multidomínio descrita.

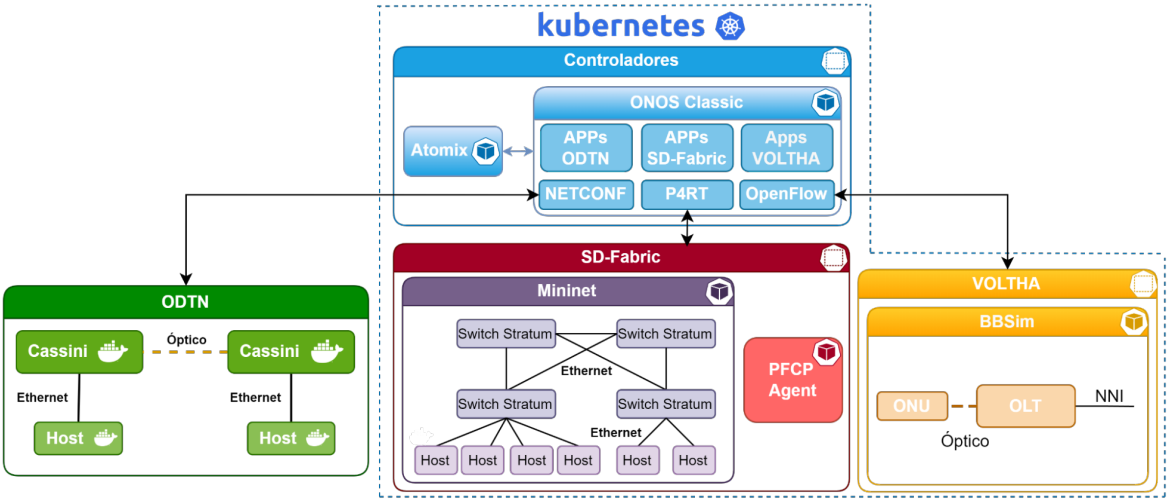


Figura 40 – Topologia implementada para a integração multidomínio.

9.2.2 Características da(s) VM(s) utilizada(s)

A implantação do ONOS Classic, juntamente com os demais projetos de cada domínio foi feita em uma Máquina Virtual (VM), contendo as seguintes características com relação a capacidade computacional:

- Processador com 8 núcleos;
- 23.5 GB de memória RAM;
- Disco rígido com 52 GB de capacidade.

9.2.3 Sistemas e Módulos de Software instalados

O ONOS e as demais iniciativas as quais o controlador constitui como uma peça importante são projetos de código aberto, logo os demais sistemas e módulos de software utilizados em paralelo seguem a mesma lógica. Portanto, para implementa-

ção do ambiente deve-se utilizar o Linux com o Ubuntu 20.04. Além disso, dentre os componentes e sistemas de software têm-se:

- Kubernetes 1.24.16;
- Docker 24.0.5;
- Helm v3.12.2.

9.2.4 Testes realizados no Pré-testbed

9.2.4.1 FTTx (VOLTHA)

Considerando a topologia descrita na subseção 9.2.1.1, os testes de implementação do domínio FTTx utilizando o projeto VOLTHA, foram testados o funcionamento da imagem do controlador ONOS Classic (ORAN-ONOS) customizada com todas as aplicações requeridas com o intuito de validar seu funcionamento. Para isso, após realizar a conexão entre os componentes do plano de controle com a infra e stack do VOLTHA foi implementado o BBSim utilizando o Helm.

Com a implementação do BBSim foi possível criar os dispositivos emulados OLTs, ONUs e NNI. Após isso, foi possível registrar utilizando o voltctl via controlador ONOS, que permite o gerenciamento dos dispositivos e seu registro de forma automática. A topologia funcionando é ilustrada na Figura 41.

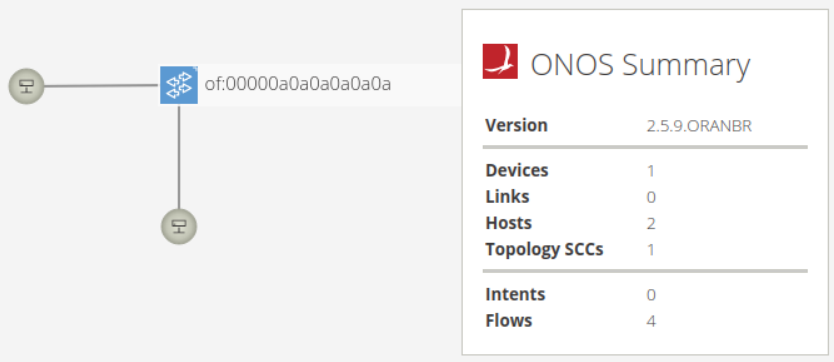


Figura 41 – Topologia do domínio FTTx.

Outra forma de apresentar os resultados obtidos está ilustrada na Figura 42, que contém as informações sobre a OLT criada pelo BBSim. Os demais dispositivos criados por esse simulador, mostrados na Figura 43, estão representados na GUI do controlador. Em que apresenta os *hosts* cadastrados, pois a ONU e a NNI são representadas e identificadas pelo controlador como pontos finais (*hosts*).

Figura 42 – Dispositivo do domínio FFTx, somente uma OLT.

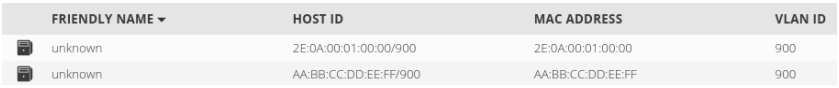


Figura 43 – Pontos finais do domínio FFTx, uma ONU e uma NNI.

9.2.4.2 P4 (SD-Fabric)

A implementação do domínio P4 para os testes, foi realizada a utilização do projeto SD-Fabric assim como definições sobre as configurações dos *switches* Stratum BMv2, os quais usam o *pipeline fabric* para permitir a programabilidade destes dispositivos, bem como, para permitir o gerenciamento destes pelo controlador ONOS. Dito isso, o ONOS passa a reconhecer os elementos da rede ao receber as informações sobre seus dispositivos e enlaces para gerenciamento, deve-se estabelecer para os Stratum, o IP de gerenciamento dos dispositivos, os quais usam o gRPC (*Google Remote Procedure Call*) para a comunicação com o controlador. Outros parâmetros para o cadastro dos dispositivos ao ONOS são o *drivers*, como stratum-bmv2, e a configuração do *pipeconf* para o *pipeline fabric*.

Para o cadastro dos enlaces entre os dispositivos, os enlaces entre os comutadores foram definidos como direto, bidirecional e persistente. A Figura 44 ilustra a topologia cadastrada, em um arquitetura *leaf-spine* e 4 *hosts* conectados no dispositivo *leaf1* e 2 no *leaf2*, totalizando 6 *hosts*. A Figura 45 apresenta os *switches* cadastrados

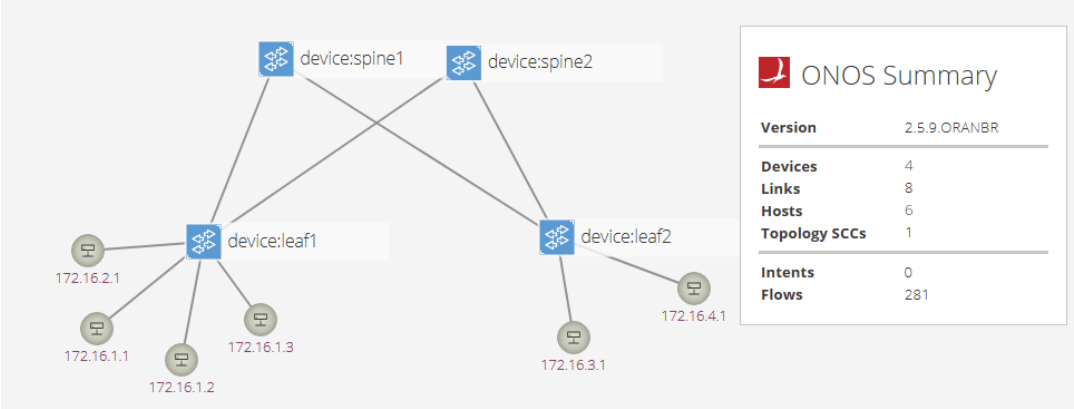


Figura 44 – Topologia do domínio p4.

e suas informações, já a Figura 46 os *links* entre os dispositivos e por fim, a Figura 47 apresenta os *hosts* e suas respectivas informações cadastradas.

	FRIENDLY NAME	DEVICE ID	MASTER	PORTS
✓	device:leaf1	device:leaf1	onos-0	9
✓	device:leaf2	device:leaf2	onos-0	9
✓	device:spine1	device:spine1	onos-0	9
✓	device:spine2	device:spine2	onos-0	9

Figura 45 – Dispositivo do domínio P4.

	PORT 1	PORT 2	TYPE
✓	device:spine2/[port2](2)	device:leaf2/2	Direct
✓	device:spine1/[port1](1)	device:leaf1/[port1](1)	Direct
✓	device:leaf2/[port1](1)	device:spine1/[port2](2)	Direct
✓	device:leaf1/[port2](2)	device:spine2/[port1](1)	Direct

Figura 46 – Enlaces do domínio P4

Figura 47 – Hosts do domínio P4

9.2.4.3 DWDM (ODTN)

Considerando a topologia descrita na subseção 9.2.1.3, que consiste em 2 cassinis e 2 *hosts*, a Figura 48 ilustra a interface gráfica do ONOS apresentando o resultado

da compatibilidade da versão desenvolvida (ORAN-ONOS) com o domínio DWDM. Além disso, a Figura 48 apresenta algumas informações sobre o controlador, como a versão do controlador e as informações sobre a topologia, incluindo o número de dispositivos, *links* e *intents*, bem como a versão do controlador.

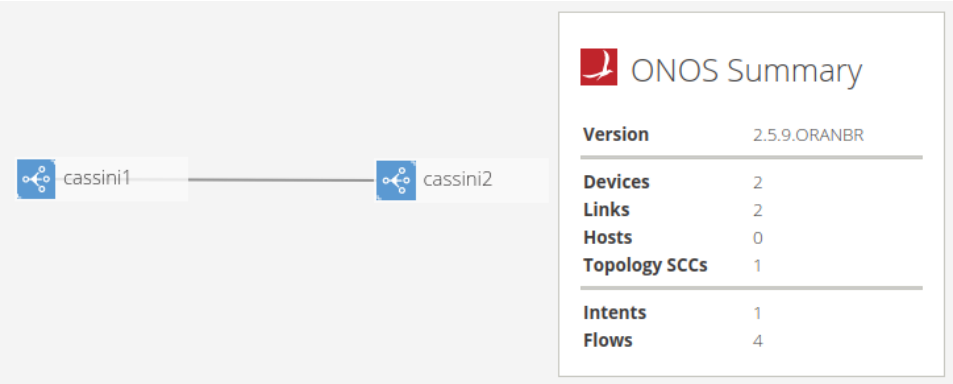


Figura 48 – Topologia do domínio DWDM.

É importante observar que a quantidade de *hosts* não foi incluída nas informações enviadas ao controlador, uma vez que não é necessária para o correto funcionamento do sistema. Portanto, esses pontos finais (*hosts*) não podem ser identificados pelo controlador, mas a topologia geral e as conexões entre dispositivos foram devidamente estabelecidas e ficaram disponíveis para o estabelecimento da conectividade e gerenciamento através do ONOS.

	FRIENDLY NAME ▲	DEVICE ID	MASTER	PORTS
✔	cassini1	netconf:172.17.0.2:830	172.17.0.6	32
✔	cassini2	netconf:172.17.0.3:830	172.17.0.6	32

(a) Dispositivo do domínio DWDM.

	PORT 1 ▼	PORT 2	TYPE
✔	netconf:172.17.0.3:830/201	netconf:172.17.0.2:830/201	Optical

(b) Enlaces do domínio DWDM.

Figura 49 – Lista de dispositivos e enlaces DWDM.

Ao falar sobre o envio dessas informações ao controlador, isto foi realizado por meio da descrição de arquivos no formato JSON, enviados por meio da REST API, em

que é definido o cadastro dos dispositivos e enlaces. No cadastro do dispositivos, algumas informações são declaradas, como o nome e o *driver* utilizado, bem como as informações para o NETCONF, indicando o IP, o mesmo do contêiner, e a porta utilizada, 830 no caso do NETCONF, isto para ambos os Cassini. Somado a isso, um arquivo JSON foi enviado contendo os dados sobre os enlaces, o qual fornece informações sobre um *link* de tipo óptico entre dois dispositivos com uma métrica de 1, que é durável e bidirecional. Algumas dessas informações podem ser observadas pela GUI do controlador, onde é demonstrada as informações sobre os dispositivos e enlaces, a Figura 49 ilustra esses dados.

APPLICATION ID	KEY	TYPE
67 : org.onosproject.optical-rest	0x0	OpticalConnectivityIntent

Figura 50 – Lista de intents do domínio DWDM.

Para mais, apenas o estabelecimento das informações sobre os *links* não garantem a conectividade da rede, portanto é necessário criar o arquivo com *intents* em forma de arquivo JSON e enviar ao controlador ONOS por meio da API REST, assim podendo informar o controlador quais interfaces ópticas estão habilitadas para transferência de pacotes, frequência e entre outros dados. Este cadastro pode ser observado na Figura 50.

9.2.4.4 Integração Multidomínio

Considerando a topologia descrita na subseção 9.2.1.4 e as implantações dos domínios FTTx, SD-Fabric e DWDM descritas nas subseções acima, realizou-se a integração do controlador SDN ONOS Classic com os 3 domínios do projeto OpenRAN@Brasil. A integração junto aos domínios levou em consideração uma série de instalações, seguindo uma determinada sequência. Primeiramente foi feita a instalação do controlador ONOS Classic dentro de um *namespace* dedicado a ele no Kubernetes. Essa instalação, considerou a definição de parâmetros do controlador, como por exemplo, aplicações a serem ativadas e módulos a serem configurados durante a inicialização.

Após a implantação do controlador, foi feita a instalação dos projetos VOLTHA e SD-Fabric dentro do cluster Kubernetes, como mostrado na Figura 51, bem como, a conexão destes projetos com o ONOS Classic. Além disso, as informações dos dispositivos transponders cassinis emulados pelo CnetLab foram enviadas ao controlador, por meio de sua *NorthBound* REST API. Ao final da integração, os dispositivos dos 3 domínios foram reconhecidos pelo controlador, sendo possível realizar o provisionamento de dispositivos e serviços via plano de controle, validando a imagem desenvolvida, como mostrado na Figura 52.

```
calico-apiserver   calico-apiserver-6d7cc79ffb-772vl                          1/1   Running   0               6h30m
calico-apiserver   calico-apiserver-6d7cc79ffb-qrbhz                          1/1   Running   0               6h30m
calico-system      calico-kube-controllers-7465dbb469-5srzc                   1/1   Running   0               6h31m
calico-system      calico-node-jg62j                                          1/1   Running   0               6h31m
calico-system      calico-typha-7d67cf786b-14vjk                              1/1   Running   0               6h31m
calico-system      csi-node-driver-khgpg                                      2/2   Running   0               6h31m
controller         onos-classic-atomix-0                                      1/1   Running   0               87m
controller         onos-classic-onos-classic-0                                1/1   Running   2 (75m ago)     87m
controller         onos-classic-onos-classic-onos-config-loader-6f489cd7d6-26m5z   1/1   Running   0               87m
infra              bbsim-sadis-server-5b5bfd8c9c-25hq5                        1/1   Running   0               6h
infra              voltha-infra-etcd-0                                        1/1   Running   0               6h
infra              voltha-infra-freeradius-7f65f684f-zcshp                    1/1   Running   0               6h
infra              voltha-infra-kafka-0                                       1/1   Running   0               6h
infra              voltha-infra-zookeeper-0                                   1/1   Running   0               6h
kube-system        coredns-57575c5f89-kh7jf                                   1/1   Running   0               6h36m
kube-system        coredns-57575c5f89-wp6rx                                   1/1   Running   0               6h36m
kube-system        etcd-victor-vm                                             1/1   Running   0               6h36m
kube-system        kube-apiserver-victor-vm                                   1/1   Running   0               6h36m
kube-system        kube-controller-manager-victor-vm                          1/1   Running   2 (44m ago)     6h36m
kube-system        kube-proxy-b618j                                           1/1   Running   0               6h36m
kube-system        kube-scheduler-victor-vm                                   1/1   Running   2 (44m ago)     6h36m
openebs            openebs-localpv-provisioner-56f4cd6457-79gfr               1/1   Running   1 (47m ago)     6h29m
openebs            openebs-ndm-49nkf                                          1/1   Running   0               6h29m
openebs            openebs-ndm-operator-6cf59c74cf-v9brb                      1/1   Running   0               6h29m
sdfabric           mininet-bcdc85944-mrdjw                                    1/1   Running   1 (114m ago)    5h7m
sdfabric           pfcp-agent-0                                               1/1   Running   1 (141m ago)    5h12m
tigera-operator    tigera-operator-959786749-67wsv                            1/1   Running   2 (44m ago)     6h31m
voltha             bbsim0-84597457f8-swn4c                                    1/1   Running   0               5h49m
voltha             voltha-voltha-adapter-openolt-5f94c8bc8f-8m4qz             1/1   Running   0               5h52m
voltha             voltha-voltha-adapter-openonu-7d6bdff9c5-p7lrk             1/1   Running   1 (5h51m ago)   5h52m
voltha             voltha-voltha-ofagent-59bdcf4d4c-p4bp4                     1/1   Running   0               5h52m
voltha             voltha-voltha-rw-core-7ff668fd79-dnkds                     1/1   Running   0               5h52m
```

Figura 51 – Projetos instalados no cluster Kubernetes.

9.3 Artigos aprovados

Durante os meses de execução das atividades relatadas nos capítulos anteriores, além da criação de um repositório público contendo todos os guias de implementação utilizados para testes foi possível realizar a submissão no **Workshop de Pesquisa Experimental da Internet do Futuro** (WPEIF) do **Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos de 2023 (SBRC)**. Este trabalho teve como título "**Redes Definidas por Software para a Orquestração de Diferentes Domínios Tecnológicos**", o qual abordou a orquestração de forma conjunta dos domí-

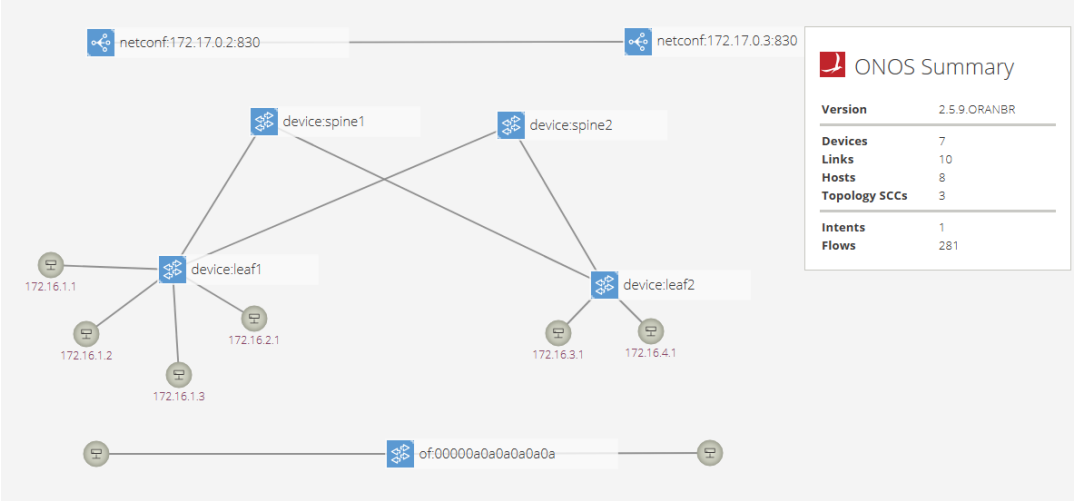


Figura 52 – Resultados dos testes de conexão multidomínio.

nios FTTx, P4 e WDM pela versão customizada do controlador ONOS Classic, criada pela equipe do laboratório GERCOM (Grupo de Estudos em Rede de Computadores e Comunicação Multimídia) da Universidade Federal do Pará (UFPA).

10 Conclusão

No contexto da preparação para o *testbed*, dada a continuação dos trabalhos realizados, alguns domínios já apresentam resultados com equipamentos do *testbed* por conta da disponibilidade dos recurso em seus respectivos domínios tecnológicos. Como descrito, em todos os domínios obtiveram avanços e novos resultados alcançados.

Este relatório apresenta o desenvolvimento do domínio P4 no capítulo 4. No capítulo 5 é apresentado o controlador inteligente RIC e sua implementação no domínio da RAN. Para integrar o domínio óptico dentro da infraestrutura, no capítulo 6 mostra a arquitetura implementada da rede de acesso. No capítulo 7, tem-se o domínio DWDM com equipamentos para a rede de transporte óptica e arquitetura SDN. Para finalizar, no capítulo 8 apresenta-se o controlador ONOS que integra todas as aplicações dos domínios tecnológicos na infraestrutura SDN.

O projeto do *testbed* está em fase de implementação de sua infraestrutura com os equipamentos adquiridos. As implementações ocorridas no pré-testbed com ambiente de emulação demonstraram resultados significativos, servindo como base para implantação do ambiente definitivo. Atualmente, com a implementação do *testbed* em ambiente real com montagem dos equipamentos físicos, além da execução técnica para cada domínio tecnológico também foca-se na integração ente os domínios de acordo com estrutura planejada para o *testbed*.

O projeto OpenRAN@Brasil está avançando de forma promissora e superando os eventuais desafios que surgem durante a implantação do ambiente em campo. A cada passo, os domínios tecnológicos estão se integrando e prospectam uma nova visão para a infraestrutura de redes conectada em regiões distintas.

11 Referências bibliográficas

1 O-RAN-Alliance. *O-RAN Working Group 3 Near-RT RIC Architecture*. Disponível em: <https://orandownloadsweb.azurewebsites.net/specifications>. Citado 2 vezes nas páginas 5 e 25.

2 CONSORTIUM, P. et al. P4runtime. *Website*, 2017. Citado na página 31.

3 (ONF), O. N. F. *Trellis*. Disponível em: <https://opennetworking.org/reference-designs/trellis/>. Citado na página 31.

4 Intel Corporation. *Atomix*. 2022. Acessado em 5 de agosto de 2023. Disponível em: <https://atomix.io/>. Citado na página 59.

5 Open Networking Foundation (ONF). *Open Disaggregated Transport Network (ODTN)*. 2019. Acessado em 5 de agosto de 2023. Disponível em: <https://opennetworking.org/wp-content/uploads/2019/04/ONF-Info-1002-ODTN-032919.pdf>. Citado na página 61.

6 Open Networking Foundation (ONF). *SEBA/VOLTHA*. 2023. Acessado em 5 de agosto de 2023. Disponível em: <https://opennetworking.org/voltha/>. Citado na página 61.

7 Open Networking Foundation (ONF). *SD-Fabric*. 2021. Acessado em 5 de agosto de 2023. Disponível em: <https://opennetworking.org/wp-content/uploads/2021/06/SD-Fabric-White-Paper-FINAL.pdf>. Citado na página 61.

8 Rede Nacional de Ensino e Pesquisa (RNP). *Emulador Óptico*. 2020. Acessado em 5 de agosto de 2023. Disponível em: <https://git.rnp.br/cnar/sdn-multicamada/emulacao/emulador-optico>. Citado na página 63.

9 Rede Nacional de Ensino e Pesquisa (RNP). *SDN Multicamada*. 2020. Acessado em 5 de agosto de 2023. Disponível em: <https://git.rnp.br/cnar/sdn-multicamada>. Citado na página 63.

10 Open Networking Foundation (ONF). *BBSim, a Broadband Simulator*. 2023. Acessado em 5 de agosto de 2023. Disponível em: <https://docs.voltha.org/master/bbsim/docs/source/index.html>. Citado na página 63.

11 Murilo Silva and Matheus Gomes and Victor Dias and Antônio Abelém. *ORAN-ONOS: OpenRAN ONOS*. 2023. Acessado em 5 de agosto de 2023. Disponível em: <https://gitlab.com/gercom-ufpa/openran/oran-onos>. Citado na página 64.

12 Open Networking Foundation (ONF). *Drivers for Cassini box with IP Infusion OC-NOS v5 supports CFP2-DCO and CFP2-ACO*. 2022. Acessado em 5 de agosto de 2023. Disponível em: <https://gerrit.onosproject.org/c/onos/+/25168>. Citado na página 70.

13 Open Networking Foundation (ONF). *SD-Fabric Helm chart - Fully virtual Installation*. 2021. Acessado em 5 de agosto de 2023. Disponível em: <https://gerrit.opencord.org/plugins/gitiles/sdfabric-helm-charts/+/HEAD/sdfabric/README.md>. Citado na página 74.

12 Histórico de versões deste documento

Data de Emissão	Versão	Descrição das Alterações Realizadas
30/09/2022	AA	Versão inicial

13 Execução e aprovação

Executado por: (CPQD)
Clériston Willian Sousa de Arruda
Isadora de Figueiredo Moreira
Joao Paulo Sales Henriques Lima
Luciano Martins
Luís Gustavo Maciel Riveros
Michelle Soares Pereira Facina
Vitalii Afanasiev

Executado por: (RNP)
Fernando Farias
Lucas Borges de Oliveira
Luiz Eduardo Folly de Campos
Ricardo Tombi

Executado por: (UNICAMP)
Christian Esteve Rothenberg

Executado por: (UNIPAMPA)
Ariel Goes de Castro

Executado por: (UFPA)
Antônio Jorge Gomes Abelém
Matheus Gomes da Costa Cordovil
Murilo Cruz da Silva
Victor Dias Leite

Executado por: (UFRJ)
Pedro Henrique Diniz da Silva

Revisado por:

_____

Luciano Martins

Aprovado por:

_____

MCTI

Data da emissão: 30/09/22